Christopher Kirkham

ID Number: 4230804

Supervisor: Henrik Nilsson

Module Code: G53IDS

2017/18

Dynamic Emotion Recognition of MIDI Music using Recurrent Neural Networks

Submitted July 2018, in partial fulfilment of

the conditions for the award of the degree **Computer Science with Artificial Intelligence (G4G7).**

Christopher Kirkham

4230804

Supervised by Dr Henrik Nilsson

School of Computer Science

University of Nottingham

I hereby declare that this dissertation is all my own work, except as indicated in the text:

Signature _____

Date _____/_____/_____

Abstract

The majority of current algorithms for music emotion recognition, while accurate, only assign emotion labels to entire pieces of music and do not represent the gradual as well as abrupt changes of emotional affect which occur within a piece. In this dissertation, a system for the dynamic recognition and labelling of emotion in MIDI music files is developed using Long Short-Term Memory neural networks. It is shown that the system can recognise these changes of emotional affect in music, and also that the use of musical chord data extracted from MIDI music can improve prediction accuracy.

Contents

1 Introduction

1.1 Background and motivation

Most current machine-learning systems for music emotion recognition apply either a single emotion label or value, or multiple labels or values, to a piece of music in its entirety [1] [2] [3]. This may be accurate enough for popular music or short classical pieces, or for purposes such as music search or playlist generation [4], but seems unsatisfactory as a detailed analysis of emotional expression in most classical music. Much of the Western classical music canon, for example, is comprised of long pieces of music which contain sections of contrasting emotion as well as gradual build-ups and transitions of emotional expression[i]; these pieces cannot be accurately categorised into a single or multiple emotion labels as a whole, nor can gradual emotional transitions be modelled by a discrete labelling system.

It is hoped that the system developed in this dissertation will go some way to allowing detailed analysis of the emotional effects of classical music given only the music (in MIDI format) as input, and that this can then be used in, for example, additional musical analysis or algorithmic music composition.

1.2 Aims and objectives

The aim of the project is to create a system which will take as input a short piece of classical music, use supervised machine learning to analyse it for emotional content, and label it accordingly.

In order to achieve this, training data must be collected and labelled in some way for emotional expression, a machine-learning architecture must be built which can learn from this data, and a way for the system to make predictions on new data must be created. Also, it is anticipated that finding the harmonies (chords) present in an input piece of music will help to improve the accuracy of emotion recognition. This will be prototyped and used if that turns out to be the case; to this end, it will also be necessary to gather and label training data for musical chords.

2 Literature review

In this section, some existing research on the topic will be considered, and preliminary thoughts presented.

2.1 Models for emotion classification

2.1.1 Overview

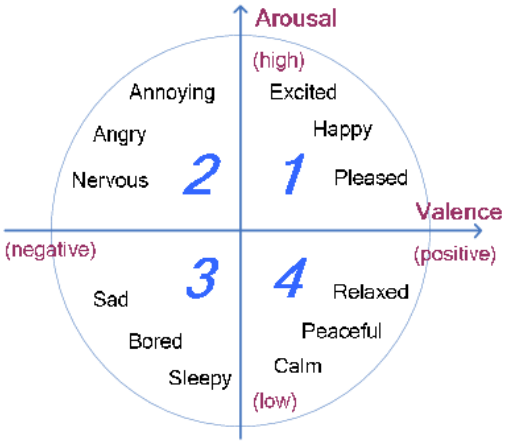
A number of models have been put forward for the task of representing and quantifying the range of human emotion, and these can broadly be categorised into lists of discrete emotional labels ("happy", "sad", "tense"…) and dimensional representations (e.g. 2D valence/arousal axes such as Thayer's model of mood, discussed later). Many of these models have their roots in psychological theory, but some have been created for other research purposes, including the study of emotion classification in music.

2.1.2 Existing models for emotion classification

This section describes several existing models for emotion classification – some with roots in psychological research, some created specifically for machine-learning tasks – and considers their respective potential advantages and disadvantages for use in this project. A final decision on which model to use will be made in the design phase.

Thayer's model of mood

Thayer's model of mood is a two-dimensional model of emotion set out in [5].



Thayer's arousal-valence emotion plane (image from [3])

Potential advantages:

- Dimensional model - ability to represent continuous changes of emotion over time – for example, slow emotional build-ups in the music – much better than with discrete labels
- Much more granular/precise than discrete labels; can represent arbitrarily precise emotion values

Potential disadvantages:

- Difficulty of placing certain emotions on a dimensional representation and/or relative to other emotions
    - e.g. sometimes hard to say whether one emotion is more of *x* attribute than another emotion
- Some emotions may seem to be at more than one place on the representation at once – e.g. nostalgia is often both happy and sad, and this is not the same as being neutral
    - A multi-label classification system would go some way to solve this problem, but would further increase time/difficulty of hand-labelling training data

Modified Thayer's model

[6] uses a version of Thayer's emotion model which divides the two dimensions into "eleven types based on Juslin's theory and Thayer's emotion model".

Simplified/categorised version of Thayer's model from [6]

Potential advantages:

- Emotion words are mapped to a dimensional model - can potentially hand-label data with discrete emotions and convert to dimensional representation for neural network, so the network learns relations between emotions

Potential disadvantages:

- Dimensional labelling will be of limited precision when converted from discrete emotions

Plutchik's wheel of emotion

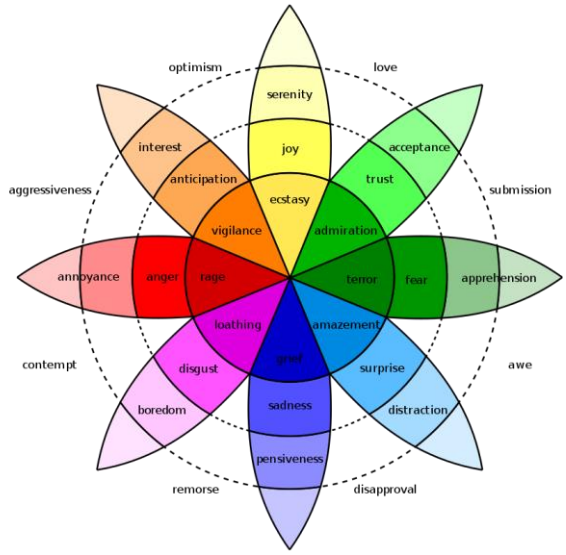
Plutchik's wheel of emotion is a 2D wheel-based emotion model proposed in [7].

Potential advantages:

- Limited number of basic emotions - potentially easy to label
- Emotion labels have degrees of intensity e.g. serenity is a milder form of joy, joy is a milder form of ecstasy
    - This may be used to allow the neural network to learn relationships between emotions, rather than treating them all as separate/unrelated

Potential disadvantages:

- Many of the labels are not words people would usually use to intuitively describe music – "trust", "distraction", "disgust"
- May make labelling more difficult if user finds lack of "right" words to intuitively describe music



Plutchik's wheel of emotion

Popular music mood categories

[8] suggests a set of labels (divided into five clusters) to categorise the mood of pieces of music. The categories were used for popular, rather than classical, music, but could well be used for this project as its labels describe moods or characters which are not specific to popular music.

Cluster1	Cluster2	Cluster3	Cluster4	Cluster5
Rowdy	Amiable/ Good natured	Literate	Witty	Volatile
Rousing		Wistful	Humorous	Fiery
Confident	Sweet	Bittersweet	Whimsical	Visceral
Boisterous	Fun	Autumnal	Wry	Aggressive
Passionate	Rollicking	Brooding	Campy	Tense/anxious
	Cheerful	Poignant	Quirky	Intense
			Silly	

Popular music mood categories from [8]

Potential advantages:

- Intuitively very expressive categories; likely easy to label (potentially easier than Plutchik's wheel)
- All categories seem like they could easily describe some music, unlike Plutchik's wheel, which contains some categories which could be considered unintuitive as musical labels.

Potential disadvantages:

- The moods are entirely discrete and do not suggest any hierarchical change of intensity, (as in Plutchik's wheel); the clusters do group related moods, but probably not in a way which would be particularly useful to a neural network for e.g. finding patterns of increasing/decreasing intensity for a mood.
- The moods in the set seem to describe characters of music rather than basic emotions. This is not necessarily a bad thing in itself so long as the music can be accurately labelled, but it does lack simpler emotions – "happy", "sad" etc. – which may be useful for classification.
- Some labels have very similar meanings (rowdy/boisterous, witty/humorous) which may make them harder to classify – could probably merge these categories with little loss of nuance.

2.2 Emotion classification using machine learning

This section considers existing attempts to classify emotion in music using machine learning, both classical music and other genres.

Detecting Emotions in Classical Music from MIDI Files

A k-nearest neighbour algorithm with attribute selection was used to classify emotion in MIDI music. Two models were used - a simpler model with four label categories, and a more complex version with twelve. Both models were based on Thayer's mood model. [1]

Music mood classification

This paper uses a simple emotion model – "happy" or "sad" – to classify pieces of music. Multiple music features are used to aid in classification; results suggest that harmony, described in the paper as "relative weighting between notes, characterized as chords or modes", increases the success rate of classification, as does musical mode. This suggests that, as first hypothesised, the inclusion of chord data may help the system to better recognise music emotions. [9]

2.3 Artificial neural networks

This section provides an overview of artificial neural networks, as well as established network architectures and libraries.

2.3.1 Overview

According to [10], "an Artificial Neural Network (ANN) is an information processing paradigm that is inspired by the way biological nervous systems, such as the brain, process information ... composed of a large number of highly interconnected processing elements (neurons) working in unison to solve specific problems ... An ANN is configured for a specific application, such as pattern recognition or data classification, through a learning process".

2.3.2 LSTM

A Long Short-Term Memory (LSTM) network is an extension of a recurrent neural network (RNN) which attempts to solve the vanishing/exploding gradient problem which hampers RNN's effectiveness. According to [11], "error signals \ flowing backwards in time" tend to either (1) blow up or (2) vanish ... Case (1) may lead to oscillating weights, while in case (2) learning to bridge long time lags takes a prohibitive amount of time, or does not work at all." The original LSTM model solves this problem by "enforcing constant error flow" through "memory cell[s]" [11].

2.3.3 Bidirectional RNN

A bidirectional RNN, described in [12], is a type of RNN in which data is trained "simultaneously in positive and negative time direction", i.e. backwards and forwards in a sequence from a given timestep. This approach may prove useful in tasks where both past and future context is important in determining the value of a label in a sequence.

2.3.4 Sequence-to-sequence (seq2seq)

A sequence-to-sequence (seq2seq) ANN is a neural network architecture which takes a sequence as input, maps that sequence to a fixed-dimension vector which represents the content of the input, and uses that vector to output a new sequence [13]. This architecture performs well in machine translation tasks [13] [14], and also showed favourable results when compared to a standard LSTM and to a generative adversarial network (GAN) in a music composition task in [15].

2.3.5 TensorFlow

TensorFlow is "an interface for expressing machine learning algorithms, and an implementation for executing such algorithms" which "has been used for conducting research and for deploying machine learning systems into production across more than a dozen areas of computer science and other fields". It aims to be a system which is "both flexible enough for quickly experimenting with new models for research purposes and sufficiently high performance and robust for production training and deployment of machine learning models". [16]

2.4 Preliminary thoughts after literature review

Emotion representation models

Of the two main approaches to emotion representation and categorisation (discrete and dimensional) considered, a dimensional approach seems like the better option. The main advantage of a dimensional model in the context of emotion recognition in sequences is the ability to model transitions between emotions - some of the most effective passages of music (especially in the Classical era) involve slow build-ups of emotion as well as abrupt contrasts or changes; it would be advantageous in terms of accuracy of emotional representation to be able to model this.

However, manually labelling the necessary training data with a dimensional model may prove taxing due to the arbitrary precision involved – indeed, the time it takes to label training data could become a deciding factor on which model to use. It may be worth considering using a discrete model, or a dimensional model with limited precision, if this will allow for the use of significantly more training data. Overall, the best emotional model to use for the purposes of this project may end up being one which provides a compromise between expressiveness and ease of manual labelling.

Artificial neural networks

Using artificial neural networks to help implement the proposed system seems like an obvious first choice. Music emotion recognition is a sequence prediction task in that a piece of music can be seen as a sequence of musical information such as notes, sounds, dynamics etc., depending on format; and that emotion label(s) must be predicted given this sequence information, either as a whole piece or per time interval. Recurrent neural networks, especially LSTMs networks, especially, have shown state-of-the-art results in sequence prediction problems including music emotion recognition and music composition, but also in fields such as neural machine translation [15] [17] [18]. Secondly, the availability of popular and well-documented neural network libraries like TensorFlow should mean that implementing a working system is relatively simple, thus allowing more time for other tasks such as data collection and prototyping different neural networks.

3 System requirements

This section details the functional and non-functional requirements of the software to be developed.

3.1 Project scope/limitations

In order to make the project feasible within the timeframe, the scope of the proposed system must be considered.

Restriction of musical style

So that the system for this project does not require an excessive amount of training data to produce good results, the training data for the system's neural nets will be restricted to music from around the Classical period of Western music: music from this period tends to be simpler harmonically and more consistent structurally than that of later eras [19]; as such, it may require less training data to learn patterns, at least for chord identification. For the same reason, the training data will also be limited to piano music – music for other instruments would not necessarily be unusable (since it's all in MIDI, there are no problems with feature extraction etc. as there may be with raw audio), but, for example, orchestral music or chamber music tends to be in a different style to piano music, so a neural net may need more data to learn from other styles.

3.2 Functional requirements

The system must meet the following functional requirements:

Input

1. User can input MIDI music files to the system
    a. Input files may be up to 12.5 seconds in length
2. Input MIDI file will be converted to a format usable by the system's neural networks

Output

The system must be able to output:

1. A comma-separated values (CSV) file containing the chord label for each timestep of input music
2. A comma-separated values (CSV) file containing the emotion label/value for each timestep of the input music
3. A graph representing the (discrete or continuous) emotion labels/values of the input music

Chord identification

The system must be able to, per timestep of music (defined in the design phase), be able to give a single chord label corresponding to the chord being played at that timestep.

Emotion identification

System will be able to, per timestep of music, be able to produce a single emotion label/value corresponding to the emotional characteristic of the music at that timestep.

User interface

The system's UI will be a simple command-line interface. The program will take as an argument the location of the input MIDI file; optional arguments will allow the user to specify whether or not to output chord and emotion CSV files.

Error handling

The system will cease operation and return an error if:

- The input is not a valid MIDI file or cannot be processed by the system into a valid format for the neural network(s)
- Output data is not successfully generated
- The user enters invalid argument(s) for the program in the command line

3.3 Non-functional requirements

Documentation

While the system is intended to be simple to use and has limited options for the user, documentation will be provided in the form of "readme" files and as help documentation for the command-line program(s).

4 Design

4.1 Overview

The system's main components are two neural networks: one LSTM network each for chord classification and emotion classification. The user's input will be run through the chord-identification network first to predict the harmonies of the input music; this will then be fed into the emotion-recognition network as features in order to improve prediction accuracy.

Note on neural network design

Neural networks are a heuristic method, and designing a neural network and fine-tuning it for optimum performance can be more of a craft than a science. Due to the potentially long training times of neural nets it would be prohibitively time-consuming to test every possible configuration of network – therefore, various significantly-different configurations will be tested and optimised during the implementation of the system so that the most effective design can be found.

4.2 Music data format

This section describes the format the processed MIDI music data (for training data and user input) will take in the neural networks, which will be consistent across all three nets in the system.

The data will include a representation of dynamics (changes in the volume of notes), as these have been shown to have an effect on the perception of emotion in music. [20] Tempo has also been shown to have an effect on the perception of emotion in music [21], so it may also be beneficial to include some representation of it in the neural network. To this end, tempo will not be explicitly represented (e.g. by a "tempo" feature); rather, timestep durations will represent a length of real time. In this way, the neural network should be able to learn the significance of note durations (and so, implicitly, tempo) on emotional effect – it also allows for seamless tempo changes within a music sequence.

With this method, the length of time represented by each timestep in a sequence must be taken into consideration. A smaller time represented per timestep will allow for greater precision of note/label timings and the representation of smaller note/label time values, while a larger time per timestep will allow longer timescales to be represented with fewer timesteps, potentially increasing the neural nets' ability to learn longer-term features of the music. For this project, a precision of 50 milliseconds per timestep has been chosen, which will produce sequences of 400 timesteps per 20 seconds. This is within the learnable sequence length posited by [11], and will allow for the representation of very small note values (a 32nd note at 120 BPM lasts just under 70 milliseconds). The rounding required to quantise notes to discrete timesteps will skew tempos slightly, but it is hoped that the timesteps are small enough to render this effect negligible.

Each timestep of music is fed into the neural networks as a vector of features of length 128. Each item represents one of the 128 possible MIDI pitches (0-127); its value (0-1) represents the velocity (loudness) with which the pitch is played at that timestep.

$$\begin{matrix} 0 \\ 1 \\ 2 \\ \dots \\ 125 \\ 126 \\ 127 \end{matrix} \begin{pmatrix} 0 \\ 0.45 \\ 0 \\ \vdots \\ 0.8 \\ 0 \\ 0.78 \end{pmatrix}$$

Example pitch vector

4.3 Chord classification

4.3.1 Overview

There is evidence that knowledge of the harmonies of a piece of music can help in emotion recognition tasks [2] [22] [23]. Therefore, a chord identification neural network will be developed – it will attempt to recognise and label the chords that are present (if any) in the input music.

4.3.2 Labelling system

While there are a huge number of different types of chords which can be identified in Western tonal music, the number of labels in this system will be restricted to those which have a harmonic function. This is both to reduce the vocabulary to allow for easier training of the network, and to place more emphasis on the harmonic functions of chords in a piece. Variations on chords that do not change their harmonic function – for example, a major 6th or minor 7th chord – will be treated as if they were a standard major or minor chord on the same root. Excluding these chords may cause the system to lose some subtlety in emotional recognition and/or expression when generating music - but such chords are rarely used in music of the classical era, so their absence will have a minimal effect on the performance of this project.

The chords which form the vocabulary for chord classification are: major, minor, seventh, diminished, and diminished 7th. These are all transposed for each of the twelve semitones in an octave with the exception of the diminished 7th chords, of which there are only three. There is also the "None" label that indicates the absence of any (full or implied) chord at the indicated time in the sequence. This results in a vocabulary of 52 discrete labels for chord classification.

Label encoding

For use in the system's neural nets, the chord labels must be encoded into a neural net-readable format of real numbers. Different encodings are possible, and may affect the performance of the network. For this system, two different one-hot-encoded orderings of chord symbols will be tested:

Ascending

In this encoding format, chord labels are arranged in a one-hot vector in ascending order by root. Different chord types (major, minor etc.) of the same root are grouped together, but otherwise there is no attempt to arrange chords by other attributes, such as harmonic relationship. The no-chord label is placed at position 0; diminished seventh chords are placed at the end of the encoding vector, as they have multiple potential harmonic meanings.

Circle-of-fifths

Chord labels are ordered by root note according to the musical "circle of fifths" [24]. By arranging the chord labels so they are closer to chords with which they have stronger harmonic relationships, and therefore which are more likely to occur in sequence in classical tonal music, it is hoped that the neural net may more easily learn which chords are likely to occur based on the surrounding chords at a given timestep. Like the ascending encoding, the no-chord label is at position 0, and diminished seventh chords are placed at the end of the encoding vector.

4.3.3 Hand-labelling system

In this section, a system for manually labelling training data with chords is described. This system is intended to facilitate hand-labelling by allowing the labeller to express timings in terms of musical measures, which should be easier than to use MIDI timesteps or real timings when working from a written score.

Labels will be entered as a .csv file in the following format: `[bar number, chord label]`. Bar numbers may include a decimal, meaning chord labels can be placed at arbitrarily precise fractions of a bar (though, during preprocessing, labels will be quantised to a certain real-time precision). The bar number represents the start of its corresponding label, which is assumed to continue until the start of the next label or the end of the file (which should also be the final bar of the corresponding piece of music). For example:

```
1, CMaj
2, DMin
3.5, GMaj
4, CMaj
```

Would represent the chord C major from the start of bar 1 until the start of bar 2, D minor from the start of bar 2 until halfway through bar 3, G major from halfway through bar 3 to the start of bar 4, and C major from bar 4 onwards.

During preprocessing, the hand-labelled chord files are converted to 1D lists of one chord (as a word) per timestep:

```
CMaj, CMaj, CMaj, CMaj, DMin, DMin, DMin, DMin, DMin, DMin, GMaj,
GMaj…
```

When these are fed into the neural network, the chord at each timestep is encoded using the chosen encoding format. It would be possible to leave the hand-labelled .csv files as they are and convert them directly into an encoded state during running of the neural network – however, the required computations are time-consuming, so it was decided to preprocess labels into the 1D list and store them prior to input, sacrificing hard-disk space for faster preprocessing.

The preprocessing of hand-labelled chord files is done using a utility script which will be described in the implementation section.

4.3.4 Task description

Chord labelling music data with the chosen labelling system is a single-label sequence classification problem. Each timestep of sequence data will be assigned a label corresponding to one chord in the vocabulary.

4.3.5 Neural network design

The following basic configurations of neural network will be tested at the implementation stage; further experimentation and hyperparameter tuning will be done on the most successful basic model.

Hidden layer(s)	**Encoding format**
RNN	Ascending
RNN	Circle-of-fifths
LSTM	Ascending
LSTM	Circle-of-fifths
Bidirectional LSTM	Ascending
Bidirectional LSTM	Circle-of-fifths

For each of these networks, the following hyperparameters will be used:

Input: 128 units

Output: Fully-connected, 52 units (number of chord labels)

Activation function: Softmax

Optimizer: Adam [25]

Epochs: 35

Batch size: 32 (choice informed by results from [26])

4.4 Emotion classification

4.4.1 Overview

It is hoped that a system which can identify, with some degree of accuracy, the emotional effects of the user's input music will have a better chance of generating music that is emotionally congruous to that music. To this end, a neural network will be trained to classify the different emotions expressed by a piece of music. The design of a labelling model which best facilitates this will be discussed, followed by a task description based on this system and the design of the neural network itself.

4.4.2 Labelling system

Based on prior research (see **3.2**), it was decided that a dimensional model for emotion recognition would likely provide the best results, as it would have the ability to model the kinds of emotional crescendos which are prevalent in classical music, as well as abrupt changes. However, it was anticipated that hand-labelling a dimensional model may be prohibitively time-consuming, necessitating a simpler design. To test this, data was manually labelled using several different labelling systems were tried prior to deciding on a final design. The conclusions are detailed below:

Tests of hand-labelling using dimensional labelling systems

Dimensional labelling using only numbers proved difficult: due to the continuous nature of the model, it was hard to judge the exact degree of any one aspect of the model. For example, a passage of music may definitely be positive in emotion, but how much? 0.5? 0.6? It was challenging to try to quantify emotion in such a way, and proved almost impossible to do in a reasonable amount of time. It was also very difficult to remain consistent in emotion ratings between pieces - more consistency may be achieved by using (averaging?) label values from multiple participants, but this would increase the workload needed, and it is still very difficult to label.

In theory, this may be a superior labelling method, as it can capture the continuously-changing nature of music's emotional effect (especially e.g. slow buildups, but also may help for shorter changes), but is impractically difficult to hand-label, at least for this project.

Tests of hand-labelling using discrete labelling systems

Discrete labelling systems were a much easier and faster method by which to hand-label training data. Depending on the system used (as predicted in **3.2.1**, the labelling system used in [8] was more intuitive to use than that of [5]), it was several times faster than purely dimensional labelling, and would allow for a much larger amount of training data to be collected in the project timescale. However, discrete labels suffer from the major disadvantage of being unable to represent gradual transitions of emotion.

Final choice of emotion labelling system

After these initial trials, it was decided that an adapted version of the Thayer-like 2-dimensional emotion model from [6] would be used in this system. The main reason for this choice is that the model seems to combine the advantages of both discrete and dimensional approaches to emotion representation: the model's discrete emotion words are mapped to a 2-dimensional space. This will allow the use of discrete labels for hand-labelling training data (likely much easier) which can then be converted to the dimensional representation for neural net training, allowing the network to take advantage of the relatedness of dimensional labels. The trade-off is that there will be limited dimensional precision, since each emotion word must be mapped to a single point on the 2D valence-arousal plane.

However, the model still allows arbitrary dimensional precision – to take advantage of this, and in order to be able to represent the gradual build-ups and transitions of emotion over time present in (classical) music, a method for hand-labelling data is presented which will allow for the linear transition between the home coordinates of different labels over time.

Adapted Thayer mood model from [6].

The coordinates corresponding to each of the emotion words in the proposed emotion model are:

Excited:	(0, 1)	Sleepy:	(0, 0)
Happy:	(1, 1)	Sad:	(-1, 0)
Pleased:	(1, 0.75)	Bored:	(-1, 0.25)
Relaxed:	(1, 0.25)	Nervous:	(-1, 0.75)
Peaceful:	(1, 0)	Angry:	(-1, 1)
Calm:	(0, 0.5)		

Simplified labelling system

In addition to the adapted Thayer model, a simplified model will also be tested. It is also based on the 11 emotion labels in the first model, but maps them to only 5 points on the arousal/valence plane:

Excited:	(1, 1)	Sleepy:	(1, 0)
Happy:	(1, 1)	Sad:	(-1, 0)
Pleased:	(1, 1)	Bored:	(-1, 0)
Relaxed:	(1, 0)	Nervous:	(-1, 1)
Peaceful:	(1, 0)	Angry:	(-1, 1)
Calm:	(0, 0.5)		

It may be that the system will perform better using this model if it cannot grasp the nuances of the first representation.

4.4.3 Hand-labelling system

The system used for manually labelling music for emotion follows the same format as the system for chord-labelling, with some added function. Each label has the format `[bar number, emotion label, modifier]` where `emotion label` is a discrete emotion word, and `modifier` is either "-" or "~". A "-" modifier instructs the preprocessing script that the coordinate corresponding to the emotion label should remain constant until the start of the next label; a "~" modifier produces a linear transition in coordinates from the start of the label coordinate to the start of the next label's coordinate. For example:

```
1, Relaxed, -
1. 5, Relaxed, ~
2, Pleasant, -
```

Would produce, assuming the standard emotion model and six timesteps per measure:

```
(1,0.25), (1,0.25), (1,0.25), (1,0.375), (1,0.5), (1,0.625),
(1,0.75), (1,0.75), (1,0.75)…
```

4.4.4 Task description

Based on the chosen labelling system, the problem of labelling music for emotion can be framed as a multivariate linear regression with two dependent variables – the x and y (valence and arousal) coordinates of the model. The neural net will need to predict these coordinates per timestep based on musical data and chord label.

4.4.5 Neural network design

Several basic configurations of neural network will be tested at the implementation stage; further tuning of hyperparameters will be done on the model which provides the best performance. In order to measure the effect of including chord data as well as music data on emotion recognition performance, tests will be run both with and without chord labels as input features; the chord encoding format which proved most successful in the chord classification network will also be used here.

Hidden layer	**Chord labels**	**Emotion label format**
RNN	None	Normal
RNN	None	Simplified
RNN	Best encoding	Normal
RNN	Best encoding	Simplified
LSTM	None	Normal
LSTM	None	Simplified
LSTM	Best encoding	Normal
LSTM	Best encoding	Simplified
Bidirectional LSTM	None	Normal
Bidirectional LSTM	None	Simplified
Bidirectional LSTM	Best encoding	Normal
Bidirectional LSTM	Best encoding	Simplified

5 Implementation

This section describes the implementation of the emotion recognition system – the tools used in the development of the system, the testing of different designs for each neural network and their final implementations, and the utility programs which were created during system development.

5.1 Training data

In total, after data augmentation, 4128 items of training/validation data were collected and labelled.

5.2 Tools used/dependencies

NumPy

Numpy [27] is "is the fundamental package for scientific computing with Python" [28]. In the project, it will be used for most data-handling operations using arrays.

pretty_midi

Pretty_midi [29] is a software package which provides a number of utilities for working with MIDI files. It will be used in this project to help preprocess MIDI files and their corresponding labels for use in the system's neural networks.

Keras

Keras is a "high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano" that "allows for easy and fast prototyping" of neural network designs. [30] It will be used in this project (with TensorFlow [16] as a backend) to implement the prototype and final designs of the neural networks in the system.

Matplotlib

Matplotlib is "a 2D graphics package used for Python for application development, interactive scripting,and publication-quality image generation across user interfaces and operating systems" [31]. It is used in the system to display emotion predictions in a graphical format.

5.3 Chord identification neural network

5.3.1 Neural network design tests

As described in the design section, several basic neural network designs will be tested for effectiveness at the chord classification task. The following table summarises the results:

Hidden layer(s)	Encoding format	Training accuracy (%)	Validation accuracy (%)
RNN	Ascending	47.72	68.57
RNN	Circle-of-fifths	48.89	70.73
LSTM	Ascending	68.02	71.53
LSTM	Circle-of-fifths	70.5	73.45
Bidirectional LSTM	Ascending	71.58	73.9
Bidirectional LSTM	Circle-of-fifths	71.99	74.09

As could be expected, basic RNNs performed significantly worse than LSTMs and bidirectional LSTMs; results were slightly better using the circle-of-fifths label encoding, suggesting that the network may indeed have been able to draw associations between chords through harmonic relation/label closeness. As the bidirectional LSTM was the best performing in these tests, it will be further refined for the final chord identification network model.

Other tests

During preliminary testing of chord-identification neural network architectures, it was considered that reducing the number of octaves of music input to the network – "squashing" the music by moving notes up or down by octaves to within a certain octave range – may help to increase prediction accuracy. However, it was found that the opposite is the case. This may be because melody notes which are not in the current chord (appogiaturas, anticipations etc.) are usually in a higher octave than some of the notes which are in the chord – especially bass notes, which tend to be part of the chordal accompaniment. It is hypothesised that, with more octaves, the network could learn to give more weight to the lower notes and less to higher ones (though what it is actually doing is likely much less simplistic); this would be harder to do with fewer octaves.

5.3.2 Final neural network model

After tests of various architectures and hyperparameters, the most effective implementation of a neural network for chord classification was found to be as follows:

Model

Input: 128 units

Hidden 1: Bidirectional LSTM, 150 units

Hidden 2: LSTM, 100 units

Output: Fully-connected, 52 units (number of chord labels)

Activation function: Softmax

Optimizer: Adam [25]

Learning rate: 0.0001

Epochs: 75

Batch size: 32

The use of a bidirectional LSTM layer increased prediction accuracy by around three percent. Musically, this makes intuitive sense: in an arpeggiated chord, the earlier notes are reinterpreted by the ear with the new information of later notes. For example, the note G followed by the note B is ambiguous. The addition of a D would form a G major chord; an E would instead form an E minor. A cell which has access to future notes as well as past will more easily be able to resolve such ambiguities.

More complex models with multiple hidden layers and/or more units per layer significantly increased training time but did not improve results even when run for more epochs.

The number of epochs to train the network for was set to 75 – after approximately this point, training set accuracy continues to increase, but validation set accuracy decreases. This suggests that the network starts to overfit to the training set, losing the ability to generalise to new data.

5.4 Emotion identification neural network

5.4.1 Neural network design tests

The following designs represent initial bases for a final neural network architecture. The most effective of these designs will be further refined by tuning hyperparameters etc.

Hidden layer	**Chord labels**	**Emotion label format**	**Training mean squared error (MSE)**	**Training mean absolute error (MAE)**	**Test MSE**	**Test MAE**
RNN	None	Normal	0.3692	0.4759	0.2550	0.3504
RNN	None	Simplified	0.4241	0.5324	0.3208	0.4174
RNN	Best encoding	Normal	0.3432	0.4666	0.2483	0.3547
RNN	Best encoding	Simplified	0.3361	0.4536	0.2481	0.3515
LSTM	None	Normal	0.3311	0.4347	0.2424	0.3504
LSTM	None	Simplified	0.3327	0.4279	0.3135	0.3880
LSTM	Best encoding	Normal	0.1708	0.2610	0.2655	0.3113
LSTM	Best encoding	Simplified	0.1597	0.2446	0.2606	0.3029
Bidirectional LSTM	None	Normal	0.3322	0.4337	0.2439	0.3352
Bidirectional LSTM	None	Simplified	0.3929	0.5086	0.2980	0.4142
Bidirectional LSTM	Best encoding	Normal	0.1259	0.2105	0.2420	0.2849
Bidirectional LSTM	Best encoding	Simplified	0.1221	0.2060	0.2423	0.2861

In all cases, after 50 epochs, training error was still converging, but test error had levelled out or was fluctuating around 0.01 to 0.02 in either direction of the final value. Therefore, this was considered a good early stopping point to test the different basic neural net designs – if the networks were run for much more time, they would likely have begun to overfit to the training data.

All of the networks which included chord labels as features performed better than those which didn't; this is in accordance with the original hypothesis that chord labels would help emotion recognition performance, and with existing research.

In both the normal and simplified labelling systems, the bidirectional LSTM performed best of the three hidden layer architectures tested, and so will be used in the final model. Somewhat surprisingly, the simplified emotion model did not perform significantly better than the normal model in the bidirectional-with-chords network – therefore, the normal model, which allows for more nuance in emotion categories, will be used for the final network architecture.

5.4.2 Final neural network model

After further experimentation on the best-performing basic emotion recognition model, a final neural network design was arrived at. Its specifications are as follows:

Input: 180 units (128 MIDI pitches; 52 one-hot encoded chord features)

Hidden 1: Bidirectional LSTM, 200 units

Hidden 2: LSTM, 100 units

Output: Fully-connected, 52 units (number of chord labels)

Activation function: tanh

Optimizer: Adam [25]

Epochs: 35

Batch size: 32

5.5 Utilities

During the development of the system, a number of small utility programs were created to speed up the process of creating, processing and augmenting training data for the system's neural networks. While these programs do not directly fulfil any of the project requirements, they have proved very useful in implementing the project, and will be included in the project source files for ease of labelling and processing new training data.

Process_music_and_chord_labels.py

This program - given an input MIDI music file, its corresponding chord label file, the tonic note of the music, and a number of musical bars n – preprocesses both music and chord labels simultaneously in the format described in the design, then outputs the corresponding chord and label sequences each of n bars in length until the end of the music.

Process_music_and_emotion_labels.py

This program, given the same input as `process_music_and_chord_labels.py` with an additional "mode" argument which tells the program to use either the normal or simplified label model, preprocesses music and emotion labels as described in the design, and outputs them in the same manner as for processing chord labels.

Transpose_music_and_labels_batch.py

This program performs data augmentation on input music and label files. Given a preprocessed music file and the corresponding preprocessed label file, it transposes both files to all twelve notes of the musical scale. It may be given an argument indicating whether chord or emotion labels are to be transposed; emotion labels are copied and renamed to correspond to the transposed music, but are not altered themselves.

6 Evaluation

6.1 Evaluation metrics

The system's final neural networks will be evaluated both in terms of their accuracy by the relevant loss functions, and by aspects of their outputs. The chord classification network is a single-label classification task, so it will be evaluated by categorical prediction accuracy; the emotion recognition neural net is a multivariate regression task, so will be evaluated by the mean-squared-error function. In addition, some outputs from the test set of each neural network will be evaluated - although not an objective evaluation method, a number of interesting observations can be made through evaluation of the neural networks' output.

6.2 Chord identification neural network

6.2.1 Network performance

The final trained chord identification neural network reached the following levels of accuracy:

Training cross-entropy: 1.1602

Training accuracy: 67.30%

Validation cross-entropy: 0.8913

Validation accuracy: 74.75%

7.2.2 Evaluation of outputs

The following are the actual and predicted chords for some of Schumann's *Kinderszenen no. 8*. For clarity, the entire label list is not shown; rather, each label is printed with the number of timesteps it appears consecutively.

Actual chords:

```
FMaj x41, F7th x7, A#Maj x26, C7 x14, FMaj x6, C7th x7, FMaj x47,
F7th x6, DMin x26
```

Predicted chords:

```
None- x14, FMaj x34, A#Maj x24, C7th x1, FMaj x3, C7th x12, FMaj x4,
C7th x8, FMaj x53, A#Maj x26, C7th x1
```

The network does get the majority of chords correct, and at least classifies them all as chords in the correct key. The "`None-`" labels at the start of the piece are understandable, looking at the score, as these correspond to a lone C note leading into the first bar. This note was hand-labelled as part of the first F major chord, but could also be interpreted as standing alone.

More worryingly, the network failed to predict the D minor chords at the end, instead labelling them as A# (Bb) major. This is only one semitone off from D minor, and both chords are in the piece's tonic key of F major, but the labelling of a minor chord as major may well have a negative effect on emotion recognition accuracy, were the network to use these predicted chords as input.

6.3 Emotion recognition neural network

6.3.1 Network performance

The final trained emotion recognition neural network reached the following levels of accuracy:

Training MSE: 0.0967

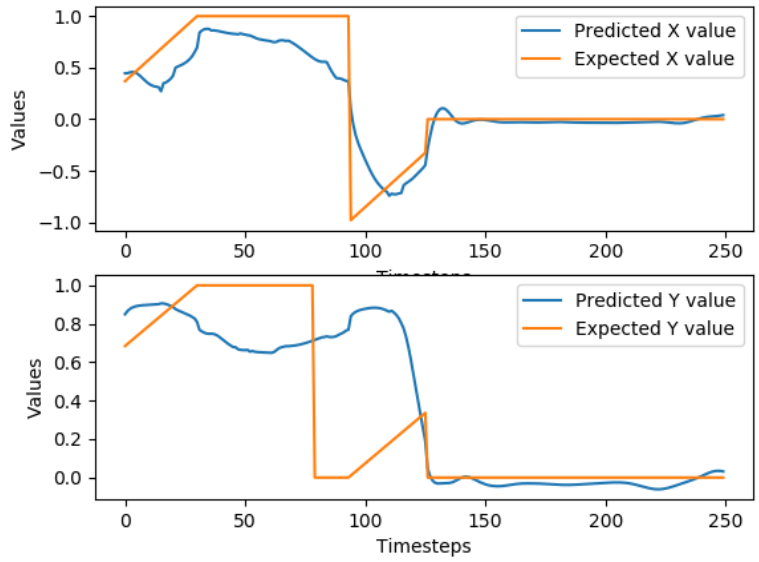
Training MAE: 0.1874

Validation MSE: 0.1610

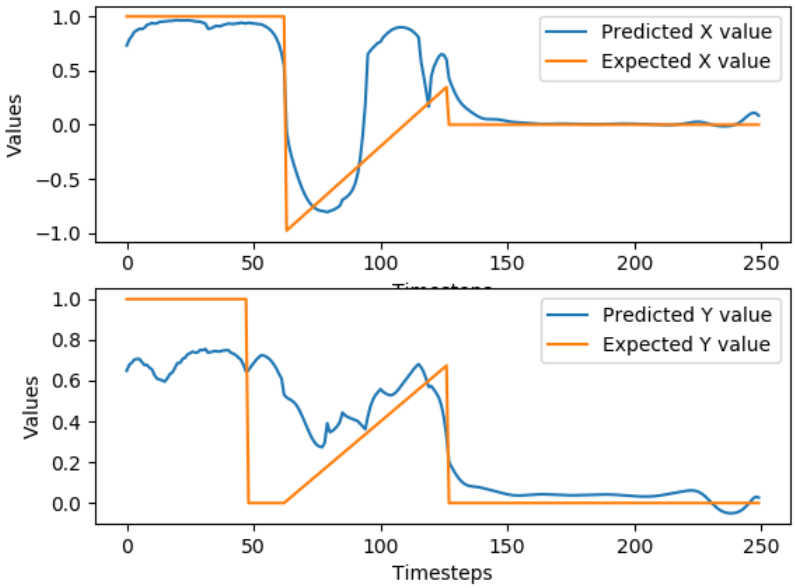
Validation MAE: 0.2440

7.3.2 Evaluation of output

A number of outputs from the networks' test set are evaluated:



In this example, the network quite accurately (at worst, +/- 0.5, 25% off) recognises the music's valence (x axis) values, and even seems to conform to the transitional (sloped) emotional passages in the sequence. The arousal (y axis), however, shows less promising results. It is reasonably accurate to the expected values up to around timestep 75; it does not at all recognise the drop in arousal at this point. From ~75 to ~125 it shows an upward gradient similar to that of the expected arousal value, suggesting that it may be recognising an increase in arousal at that point, if not the actual arousal level, but this is not necessarily the case.



Again, the network recognises shifts in valence somewhat well, although it performs worse with the transitional section than hoped. The arousal predictions, while inaccurate at first, follows the transitional section quite well from timesteps ~75 onwards.

See **appendix B** for more examples of emotion recognition performance.

7 Reflections

7.1 Software design

Emotion labelling model

Significant time was spent deciding on the model for emotion recognition. Hand-labelling the training data with discrete words then converting them to a dimensional representation during preprocessing allowed for easy testing of different models based on the same set of words (the normal and simplified models in the system), the only change necessary being the dimension values for preprocessing. While the actual results could be improved, I think I made the correct choice of labelling model.

Training data

Gathering and hand-labelling the training data necessary for the system's neural networks turned out to be one of the most time-consuming parts of the project. Even with a restricted vocabulary of chords and a relatively simple emotion model, I estimate that labelling a piece of music for both chords and emotions took two to three hours on average. This process included listening to the piece in question at least twice while following the score to get a sense of the music's characteristics, hand-labelling the chords and emotions on a paper score (the chords were often labelled from the score alone; the emotion labelling was done while listening to the music, and often required several listens), and inputting these labels into a text editor in the correct format for preprocessing.

While, after data augmentation, the dataset size numbered over four thousand sequences, this is still tiny compared to well-known datasets such as MNIST [32], which contains around 60000 examples. The dataset allowed for a successful enough result, and a proof-of-concept for certain ideas I had for the project – specifically, the use of chord identification to improve the effectiveness of emotion recognition.

Flaws in training data

Some of the pieces used in the training data had several repeated sections. While repetition and variation on a theme in music is normal, long repeated sections, such as in the third movement of Schubert's Sonata in Bb, D.946, caused some sequences to be the same or very similar to others from the same piece. This may have resulted in some overfitting to these sequences, as it is essentially giving the same data more prominence within the networks. This is, again, something which the use of more training data could alleviate, if it does affect the results of chord or emotion recognition.

7.2 Project management

Timescale and project scope

A timescale for each aspect of the project was drawn up before work began (see appendix *C* – project Gantt chart). However, ongoing non-project-related issues on my part resulted in some significant setbacks, so the initial timescale is no longer representative of how long each aspect of the project took. Indeed, the initial project proposal was to create a music generation system based on the emotion recognition model created in this project – and this is still something which I think would be interesting - but the setbacks meant that this was no longer feasible in the timeframe.

Version control

Version control (Git [33]) was utilised throughout the project; commits were regular, though no strict commit schedule or version control practice e.g. continuous integration was followed. My use of version control was adequate for this project, but a project with multiple developers would need a stricter approach, as well as better use of Git's branching features.

Research

Research was collated methodically before development started, which proved very helpful during the literature review stage and when referring back to existing research during the design and implementation stages of the project.

7.3 Conclusions

Based on the results shown by the system's neural networks, it could be said that the project has been a moderate success. The chord identification neural network mostly performs satisfactorily. The emotion recognition network shows definite room for improvement, but as a proof-of-concept – to recognise and model abrupt changes, build-ups and other transitions of emotional expression in a piece of classical music – it is somewhat successful.

7.4 Further research/improvements

Neural network design

Both the chord identification and emotion recognition neural networks performed best with only one hidden bidirectional LSTM layer. It may be that, given more training data, a more complex model will be able to extract nuances from the data which the simpler model cannot, but that currently there is not enough data for a more complex model to improve results.

Longer sequences

For this project, the sequence length of the input was limited to 250 timesteps, or 12.5 seconds at the chosen 50 milliseconds per timestep. This was done to allow pieces of music to be split into smaller sections for training, reducing the amount of training data necessary to be collected and labelled. Ideally, entire pieces, or at least high-level sections of a piece (e.g. the first subject of a sonata-allegro) will be able to be labelled as a whole. Up to a certain point, the input sequence length could be increased while retaining good results by increasing the amount of training data with longer sequences available, but for very long sequences - >1000 timesteps is a likely upper limit, as posited by [11] – a different neural network or other machine learning model may be required. A sequence-to-sequence model (considered in **2.3.3**) with a mechanism such as Bahdanau attention [18] may be able to retain information over long time gaps which can alter the emotions impressed upon a listener, for example the reoccurrence and/or variation of themes which appeared earlier in a piece, better than a simple recurrent neural net.

8 Acknowledgements

I would like to give thanks to Henrik Nilsson, who has been extremely helpful and eternally patient. I would also like to thank Gail Hopkins, Carole East and the university's support network, without whose help and guidance I could not have finished this work, let alone the entire degree course.

9 Bibliography

[1] J. Grekow and Z. W. Raś, "Detecting Emotions in Classical Music from MIDI Files," 2009. [Online]. Available: https://www.researchgate.net/publication/226269425_Detecting_Emotions_in_Classical_Music_from_MIDI_Files.

[2] C. e. al., "Music emotion recognition using chord progressions," 2016. [Online]. Available: https://ieeexplore.ieee.org/document/7844628/.

[3] y.-h. Yang, Y.-C. Lin and H. Chen, "Music Emotion Classification: A Regression Approach," August 2007. [Online]. Available: https://www.researchgate.net/publication/4266636_Music_Emotion_Classification_A_Regression_Approach.

[4] J. A. F. Fernandes, "Automatic Playlist Generation via Music Mood Analysis," 2010. [Online]. Available: http://mir.dei.uc.pt/pdf/Theses/MOODetector/Fernandes%20MSc%20Thesis%202010.pdf.

[5] R. E. Thayer, The Biopsychology of Mood and Arousal, New York: Oxford University Press, 1989.

[6] H. e. al., "SMERS: MUSIC EMOTION RECOGNITION USING SUPPORT VECTOR REGRESSION," 2009. [Online]. Available: https://www.cs.cmu.edu/~rbd/papers/emotion-ismir-09.pdf.

[7] R. Plutchik, "Emotions: A general psychoevolutionary theory," 1979. [Online]. Available: https://www.researchgate.net/publication/243745753_Emotions_A_general_psychoevolutionary_theory.

[8] X. Hu and J. S. Downie, "EXPLORING MOOD METADATA: RELATIONSHIPS WITH GENRE, ARTIST AND USAGE METADATA," 2007. [Online]. Available: https://pdfs.semanticscholar.org/4421/83b2518696b0ce86e818509373990c7f69c0.pdf.

[9] J. Padial and A. Goel, "Music Mood Classification," [Online]. Available: http://cs229.stanford.edu/proj2011/GoelPadial-MusicMoodClassification.pdf.

[10] S. B. Maind and P. Wankar, "Research Paper on Basic of Artificial Neural Network," January 2014. [Online]. Available: http://www.ijritcc.org/download/Research%20Paper%20on%20Basic%20of%20Artificial%20Neural%20Network.pdf.

[11] H. e. al., "Long Short-term Memory," 1997. [Online]. Available: http://www.bioinf.jku.at/publications/older/2604.pdf.

[1 2] M. Schuster and K. K. Paliwal, “Bidirectional Recurrent Neural Networks,” 1997. [Online]. Available: https://pdfs.semanticscholar.org/4b80/89bc9b49f84de43acc2eb8900035f7d492b2.pdf.

[1 3] I. Sutskever, O. Vinyals and Q. V. Le, “Sequence to Sequence Learning with Neural Networks,” 2014. [Online]. Available: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf.

[1 4] C. e. al., “Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation,” 2014. [Online]. Available: http://emnlp2014.org/papers/pdf/EMNLP2014179.pdf.

[1 5] N. Agarwala, Y. Inoue and A. Sly, “Music Composition using Recurrent Neural Networks,” [Online]. Available: https://web.stanford.edu/class/cs224n/reports/2762076.pdf.

[1 6] A. e. al, “TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems,” 9 November 2015. [Online]. Available: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45166.pdf.

[1 7] D. Eck and J. Lapalme, “Learning Musical Structure Directly from Sequences of Music,” [Online]. Available: http://www.iro.umontreal.ca/~eckdoug/papers/tr1300.pdf.

[1 8] D. Bahdanau, K. Cho and Y. Bengio, “NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE,” 2015. [Online]. Available: https://arxiv.org/pdf/1409.0473.pdf.

[1 9] University of St. Thomas, “The Classical Period,” [Online]. Available: https://www.stthomas.edu/media/selimcenter/pdf/TheClassicalPeriod.pdf.

[2 0] S. B. Kamenetsky, D. S. Hill and S. Trehub, “Effect of Tempo and Dynamics on the Perception of Emotion in Music,” October 1997. [Online]. Available: https://www.researchgate.net/publication/247733529_Effect_of_Tempo_and_Dynamics_on_the_Perception_of_Emotion_in_Music.

[2 1] A. Fernández-Sotos, A. Fernández-Caballero and J. M. Latorre, “Influence of Tempo and Rhythmic Unit in Musical Emotion Regulation,” 3 August 2016. [Online]. Available: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4971092/.

[2 2] C. e. al., “AUTOMATIC CHORD RECOGNITION FOR MUSIC CLASSIFICATION AND RETRIEVAL,” 2008. [Online]. Available: https://users.ece.cmu.edu/~hengtzec/papers/icme08_chord.pdf.

[2 3] D. Bakker and F. Martin, “Musical chords and emotion: major and minor triads are processed for emotion.,” March 2015. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/24957406.

[2 4] www.dummies.com, “The Circle of Fifths: a Brief History,” [Online]. Available: https://www.dummies.com/art-center/music/the-circle-of-fifths-a-brief-history/.

[2 5] D. P. Kingma and J. Lei Ba, “ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION,” 2015. [Online]. Available: https://arxiv.org/pdf/1412.6980.pdf.

[26] K. e. al., “ON LARGE-BATCH TRAINING FOR DEEP LEARNING: GENERALIZATION GAP AND SHARP MINIMA,” 2017. [Online]. Available: https://arxiv.org/pdf/1609.04836.pdf.

[27] O. Travis E, “A guide to NumPy,” Trelgol Publishing, 2006.

[28] NumPy, “NumPy,” [Online]. Available: numpy.org.

[29] C. Raffel and D. P. W. Ellis, “Intuitive Analysis, Creation and Manipulation of MIDI Data with pretty_midi,” 2014. [Online]. Available: http://colinraffel.com/publications/ismir2014intuitive.pdf.

[30] keras-team, “GitHub - keras-team/keras: Deep Learning for humans,” [Online]. Available: https://github.com/keras-team/keras.

[31] H. e. al., “Matplotlib: A 2D graphics environment,” *Computing in Science & Engineering,* vol. 9, no. 3, pp. 90-95, 2007.

[32] Y. LeCunn, C. Cortes and C. J. Burges, “THE MNIST DATABASE of handwritten digits,” 1998. [Online]. Available: http://yann.lecun.com/exdb/mnist/.

[33] “Git,” [Online]. Available: https://git-scm.com/.

[34] Aiva Technologies, “Composing the music of the future – Aiva Technologies – Medium,” 24 September 2016. [Online]. Available: https://medium.com/@aivatech/composing-the-music-of-the-future-4af560603988.

[35] Google , “Magenta,” [Online]. Available: https://magenta.tensorflow.org/.

[36] O. e. al., “Learning to Create Piano Performances,” 2017. [Online]. Available: https://nips2017creativity.github.io/doc/Learning_Piano.pdf.

[37] H. e. al., “ONSETS AND FRAMES: DUAL-OBJECTIVE PIANO TRANSCRIPTION,” 30 October 2017. [Online]. Available: https://arxiv.org/pdf/1710.11153.pdf.

[38] A. e. al., “AUDIO DEEPDREAM: OPTIMIZING RAW AUDIO WITH CONVOLUTIONAL NETWORKS.,” 2016. [Online]. Available: https://pdfs.semanticscholar.org/7440/725d3a8d4b766ca5e612443774d32dc31845.pdf.

[39] yourstory.com, “This AI startup has achieved a human feat – composing music,” 10 October 2017. [Online]. Available: https://yourstory.com/2017/10/ai-startup-achieved-human-feat-composing-music/.

[40] J. e. al., “Generating Music by Fine-Tuning Recurrent Neural Networks with Reinforcement Learning,” 2016. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/45871.pdf.

[4 P. e. al., “On the difficulty of training Recurrent Neural Networks,” 2013. [Online]. Available:
1] https://arxiv.org/pdf/1211.5063.pdf .

Appendices

Appendix A Chord identification encodings

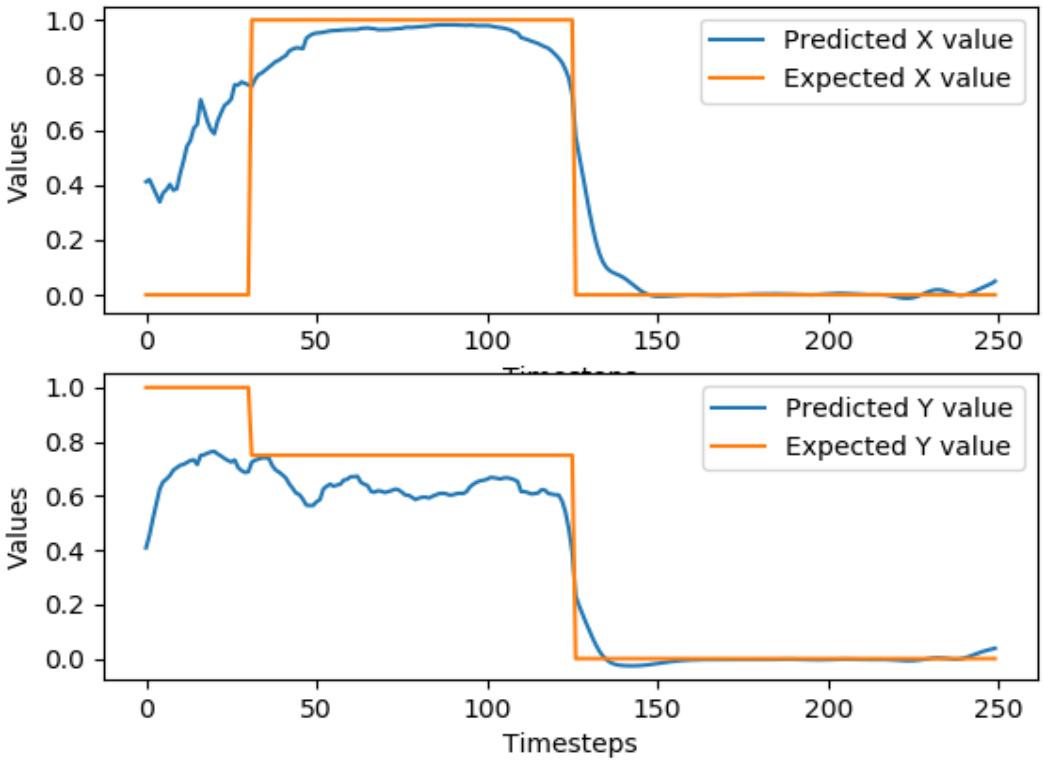
Ascending encoding

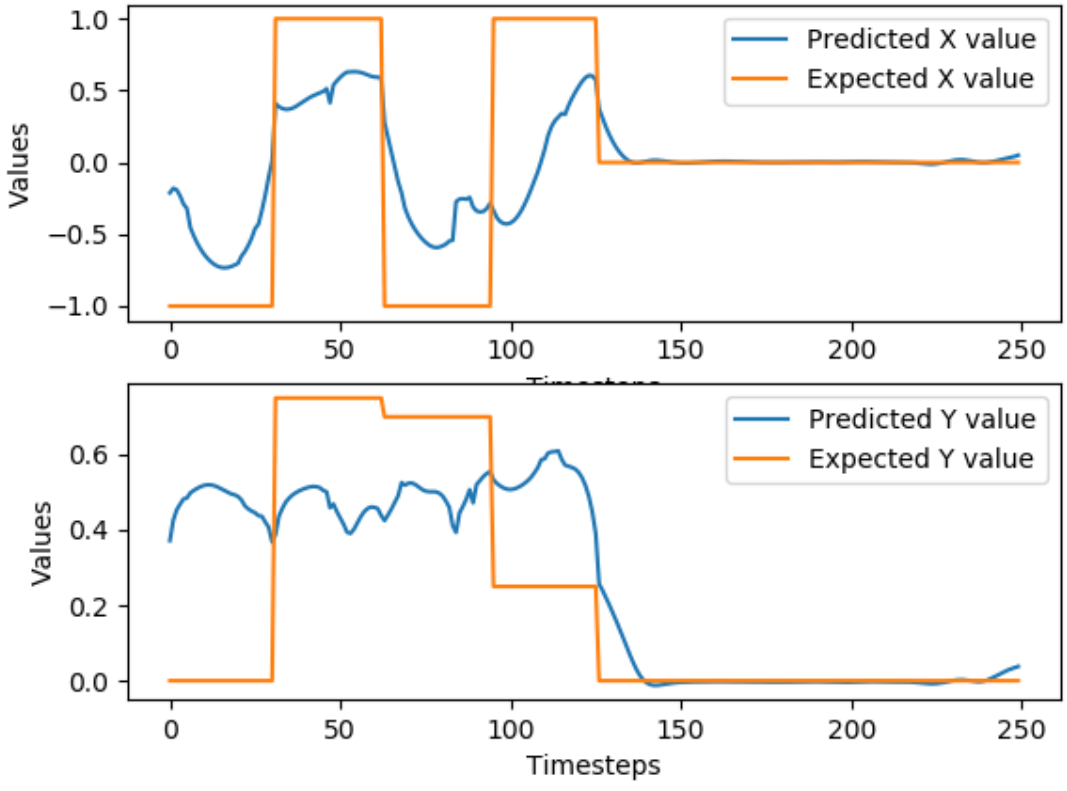
```
"None-", "CMaj", "CMin", "C7th", "CDim", "C#Maj", "C#Min", "C#7th",
"C#Dim", "DMaj", "DMin", "D7th", "DDim", "D#Maj", "D#Min", "D#7th",
"D#Dim", "EMaj", "EMin", "E7th", "EDim", "FMaj", "FMin", "F7th",
"FDim", "F#Maj", "F#Min", "F#7th", "F#Dim", "GMaj", "GMin", "G7th",
"GDim", "G#Maj", "G#Min", "G#7th", "G#Dim", "AMaj", "AMin", "A7th",
"ADim","A#Maj","A#Min","A#7th","A#Dim","BMaj","BMin","B7th","BDim","
C-/D#-/F#-/A-Dim7","C#-/E-/G-/A#-Dim7","D-/F-/G#-/B-Dim7"
```

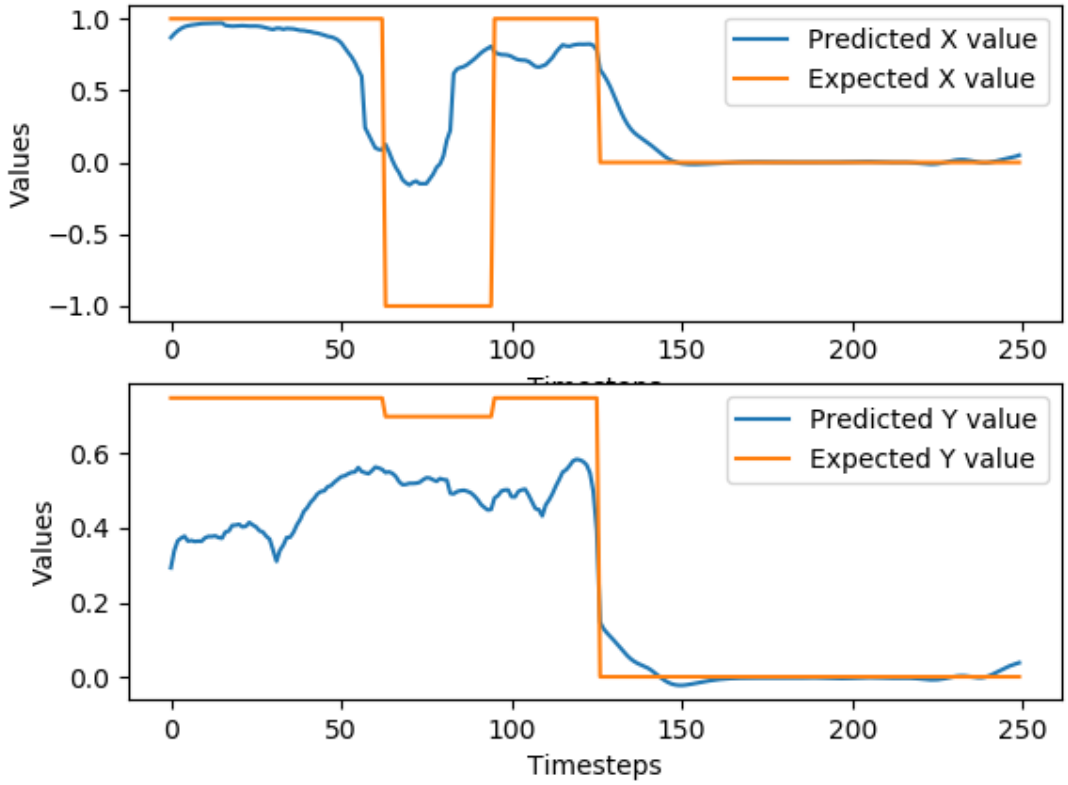
Circle-of-fifths encoding

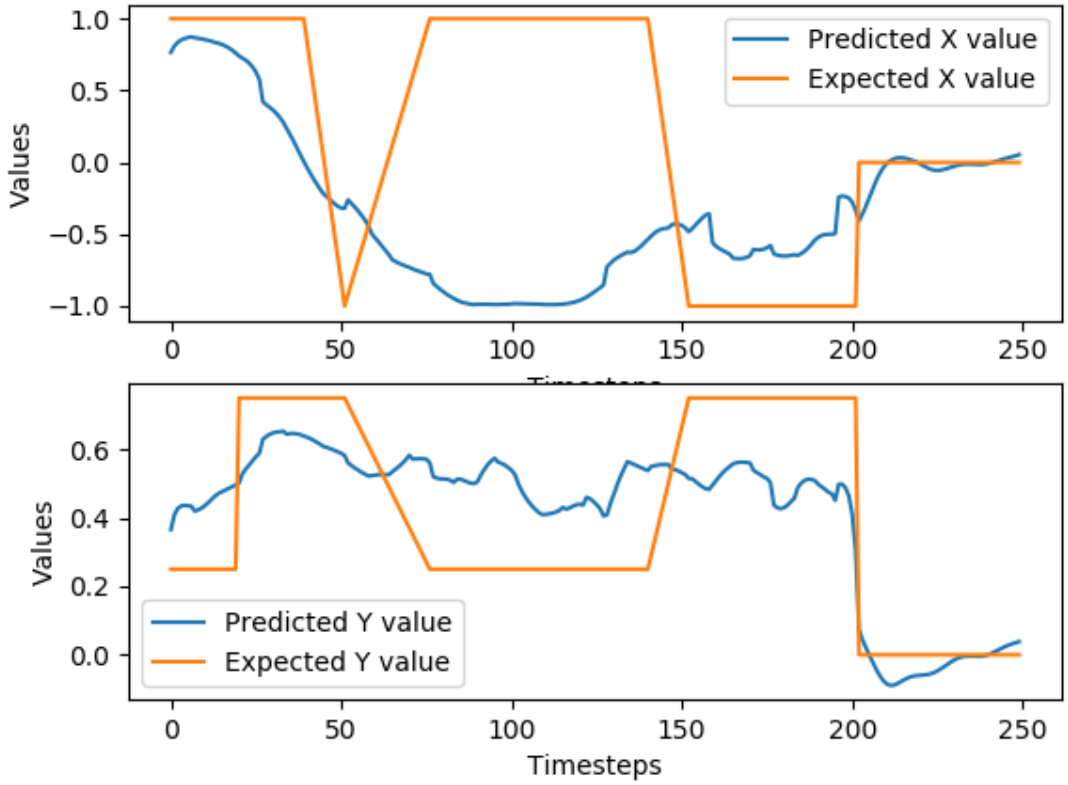
```
"None-", "F#7th", "A#Dim", "F#Maj", "F#Min", "C#7th", "FDim",
"C#Maj", "C#Min", "G#7th", "CDim", "G#Maj", "G#Min", "D#7th",
"GDim", "D#Maj", "D#Min", "A#7th", "DDim", "A#Maj", "A#Min", "F7th",
"ADim", "FMaj", "FMin", "C7th", "EDim", "CMaj", "CMin", "G7th",
"BDim", "GMaj", "GMin", "D7th", "F#Dim", "DMaj", "DMin", "A7th",
"C#Dim", "AMaj", "AMin", "E7th", "G#Dim", "EMaj", "EMin", "B7th",
"D#Dim","BMaj","BMin","C-/D#-/F#-/A-Dim7",
"C#-/E-/G-/A#-Dim7","D-/F-/G#-/B-Dim7"
```

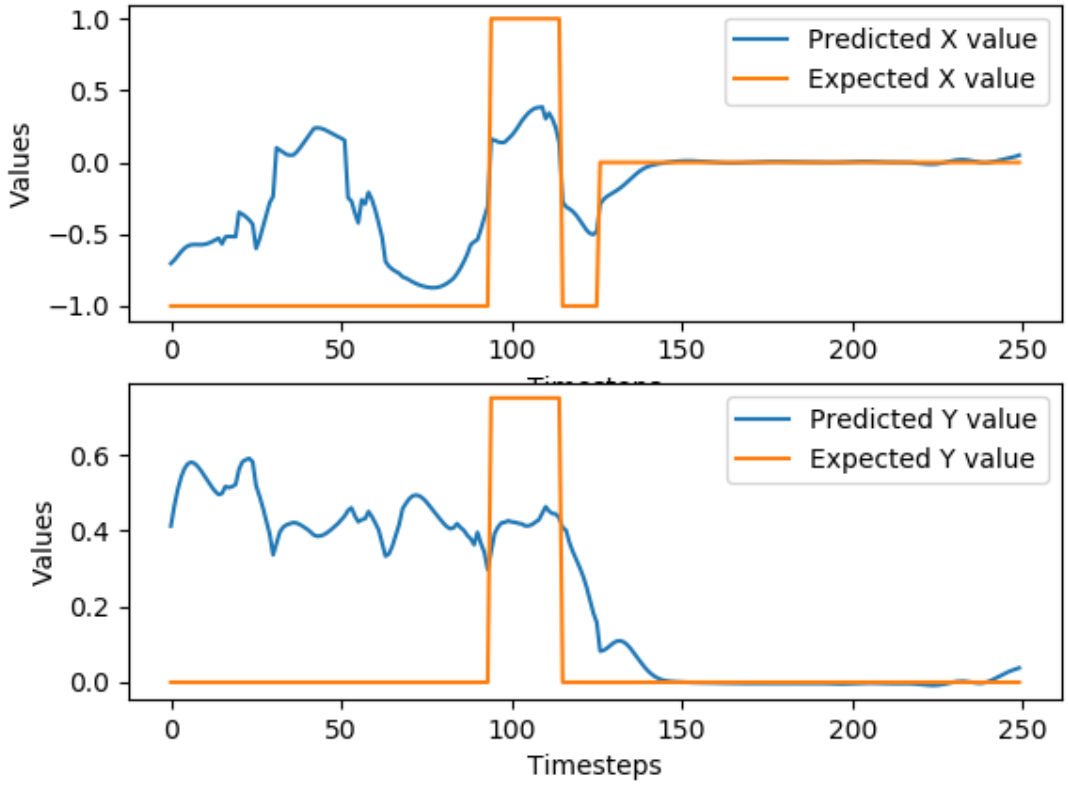
Appendix B Emotion recognition results examples






Predicted X value
Expected X value
Values
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250
Predicted Y value
Expected Y value
Values
0.6
0.4
0.2
0.0
0
50
100
150
200
250
Timesteps


Predicted X value
Expected X value
Values
1.0
0.5
0.0
−0.5
−1.0
0
50
100
150
200
250
Predicted Y value
Expected Y value
Values
0.6
0.4
0.2
0.0
0
50
100
150
200
250
Timesteps



[i] As an example of a piece of music which contains long sections of contrasting emotion as well as both abrupt and gradual changes of emotional expression within a section, see Beethoven's *Piano Sonata No. 5 in C minor, Op. 10, No. 1, Mvmt. 1.*