HERIOT WATT UNIVERSITY

MUSIC COMPOSITION USING DEEP LEARNING

Fourth Year Dissertation

MEng Software Engineering

Supervisor: Ioannis Konstas
Second Reader: Mehran Sharghi

George Malcolm Hughes
H00245407
gmh1@hw.ac.uk

1 Declaration

I, George Hughes, confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: George Hughes

Date: 05/11/19

2 ABSTRACT

With the rise of big data and extremely high processing power, the resulting accessibility of large datasets and ability to quickly process them has caused deep learning to enter the mainstream of computer science, allowing for solutions to highly complex problems to be discovered and approximated in many different pursuits.

Recently, a number of studies have been conducted on the capabilities of recurrent neural networks – a type of deep learning algorithm – to generate music, treating it as a natural language and applying techniques as such. The promising results give rise to interesting ideas about the future of music composition, potentially providing an answer to the universally abhorred condition of artist's block by suggesting ideas to the frustrated musician.

This project explored novel ways that neural networks can be used to generate monophonic music by employing two different approaches to network architecture. The first approach was to use multitask learning, giving the network a main task – predicting the next music note in a melody – and auxiliary tasks – one for following structure and one for comprehending harmony. The other approach investigated the potential use of hierarchical neural networks, dedicating each level of the hierarchy to the different hierarchical levels of music, such as the note level and the bar level.

The music generated using multitask learning models was initially evaluated using statistical perplexity then, upon finding an optimising design and hyperparameters, by human participants for their sentiments, finally comparing these results to those of a baseline recurrent neural network using the same evaluation strategy. Human participants also evaluated human-composed music to create a target for the models and second point of comparison.

3 CONTENTS

4 INTRODUCTION

It is often said that music is a universal language (Mehr, et al., 2018), and it is also true that the idea of music and maths being intertwined is a common one (Vaughn, 2000). If music is both a language and possesses mathematical attributes, then a computer can surely model and reproduce the patterns that can be found in the structure, harmony and rhythm of a composition for the sake of driving the music-making process with automation.

In the other corner, Deep Learning is a constantly evolving technology which has applications in sequence pattern recognition, making it useful to the field of natural language processing (Goldberg, 2017). It involves training weights of connections in a network structure to approximate a function by using large data sets which contain pairs of inputs and expected outputs – with sequence prediction being performed by having an input of previous tokens in the sequence and an output of the next token, which gives the network the goal of predicting the next item in an input sequence. This could be applied to monophonic music, which is of a sequential nature, by predicting the next note in a melody given some previous notes; from this we have a tool to assist in the composition process.

It was chosen to explore neural networks to fill a gap in the literature, since there is little background available for the architectures examined here and research is particularly lacking in their application to music even though they appear to be suited to the nature of music.

4.1 AIMS

The overall aim of this project was to explore the use of different deep learning techniques for generating original, monophonic Irish Folk music. The genre was chosen due to the simplicity of its structure, its reliance on monophonic melodies and the availability of a large dataset of music in the style[1]. The paper aims to continue previous work using this dataset. (Sturm, et al., 2016)

The music was generated by incorporating Natural Language Processing techniques: treating music as a textual language, and Recurrent Neural Networks: defining goals and training with datasets of music, with the generated music being based on learned predictions by the network.

The two deep learning techniques that this project aimed to focus on demonstrating were multitask learning and hierarchical neural networks, with the goal of showing that these can be effectively applied to music as a natural language by comparing their results to those of a baseline recurrent neural network.

To initially rate the models before exposure to human participants, a measure of perplexity was used to retrieve statistical knowledge about how well they predicted music from the training data; models with lower perplexities were used to demonstrate that useful pattern detection had been acquired. A definition for perplexity is given in section 7.5.1.

One specific goal was for the generated music to sound like it was composed by a human; to test this, a survey was conducted which asked human participants for their sentiments on pieces of music after listening, also gathering results for human-

[1] Data available at https://thesession.org/

composed music from the dataset to compare against. This also provided insight on how much participants enjoyed listening to the music.

4.2 Hypotheses

Hypotheses questions that this project aimed to answer are formalised below.

- If a multi-task learning neural network is trained on a dataset of monophonic Irish folk music, then:
    - It can be used to create music whose performance by the measure of positive to negative experience on a Likert scale will exceed that of a standard LSTM RNN.
    - It can be used to create music whose performance by the measure of positive to negative experience on a Likert scale will match that of human composed music.
- *(Optional)* If a hierarchical neural network is trained on a dataset of monophonic Irish folk music, then:
    - It can be used to create music whose performance by the measure of positive to negative experience on a Likert scale will exceed that of a standard LSTM RNN.
    - It can be used to create music whose performance by the measure of positive to negative experience on a Likert scale will match that of human composed music.

4.3 Organisation

Aside from introducing the project and its objectives, this report contains sections for:

- Providing a background of technical concepts required in section 5,
- Discussing and reviewing relevant technical literature in section 6,
- How the system was designed and developed including discussion of important design choices in section 7,
- An analysis and discussion of results from evaluation in section 8,
- And lastly, a conclusion covering project successes and limitations in section 9.

5 Background

Below, a breakdown of all relevant technical concepts is given, relating each concept to this project.

5.1 Language Modelling

Language Modelling is a subset of Natural Language Processing, encompassing various techniques and processes which attempt to formalise natural languages and make predictions about the sentences within them.

A natural language is one which arises without being consciously designed; languages have a *vocabulary*, which is a set of all *symbols* (or *tokens*) which comprise the language, and *sentences*, which are strings of symbols from the language's vocabulary.

In this project, music was treated as a natural language, with symbols and sentences being extracted from the music training data.

5.2 Neural Networks

A *neural network* can be thought of as an architecture which models the biology of a brain processing information (in a massively simplified manner). It can be thought of as a graph, shown in Figure 1, but where the nodes are called *neurons* and the connections are called *weights*.

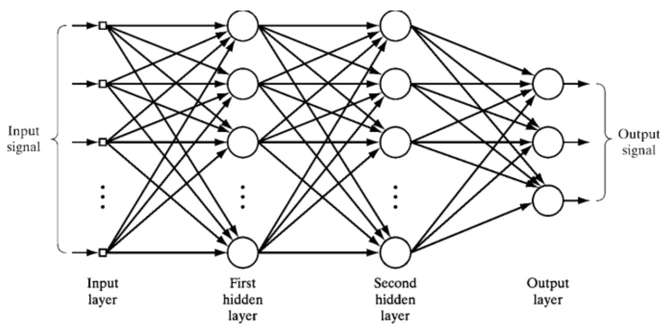


FIGURE 1 – MULTILAYER PERCEPTRON WITH THREE LAYERS (HAYKIN, 2009)

A neural network has parameters consisting of weights and *biases*; formally, it consists of a matrix of weights W and a bias b for each *layer*:

$$W^L = \begin{pmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,m} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,m} \\ \vdots & \vdots & \ddots & \vdots \\ w_{n,1} & w_{n,2} & \cdots & w_{n,m} \end{pmatrix} w_{i,j} \epsilon \mathbb{R}$$

Where L is the layer and $w_{i,j}$ is the weight from neuron i to neuron j.

Given an input matrix $x = \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}$ for a layer, an output matrix y can be formed:

$$y = \begin{pmatrix} 1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix}$$

$$y_j = \varphi(v_j)$$

$$v_j = \sum_{i=1}^{n} x_i w_{i,j} + b$$

Each y_n is considered to be the *activation signal* of neuron n and the activation signals in a layer y become the input for the next layer.

The specifics of neural networks relevant to this project are covered in the following sections.

5.2.1 ACTIVATION FUNCTIONS

An *activation function* $\varphi(v)$ is a differentiable function which calculates the output of a neuron with the purpose of limiting the range of values which can be outputted (hence the alternative name: *squashing function*).

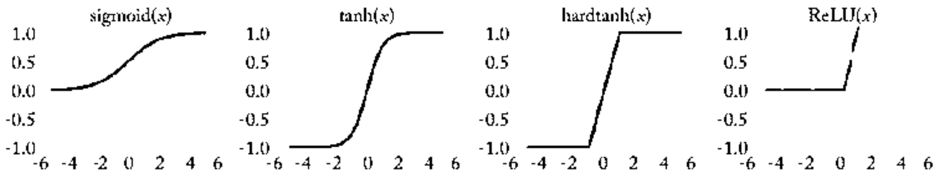


FIGURE 2 – ACTIVATION FUNCTIONS AND THEIR DERIVATIVES (GOLDBERG, 2017)

With a variety of different activation functions available, consideration should be given with respect to the circumstances. Figure 2 shows the various functions and their derivatives; their derivatives being necessary for backpropagation (this will be returned to in the next section).

5.2.2 MULTI-LAYER PERCEPTRON

The *multi-layer perceptron* is a neural network structure, consisting of at least two layers (including one hidden layer and one output layer), which can be used for classification of non-linearly separable patterns. (Haykin, 2009)

A *hidden layer* of a neural network is one which is hidden from but connects the input and output nodes. Hidden layers allow for more complex patterns to emerge (Figure 3).

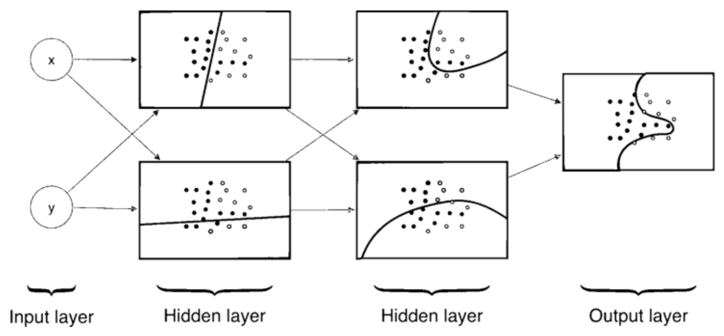


FIGURE 3 – VISUALISATION OF A NEURAL NETWORK SOLVING COMPLEX CLASSIFICATION PROBLEMS (SERRANO, 2016)

The multi-layer perceptron obtains its ability to represent non-linear functions due to using arbitrary, but differentiable, non-linear activation functions at each hidden layer.

A network is referred to as being *fully connected* if all neurons in one layer are connected to all neurons in the next layer; multi-layer perceptrons are always fully connected, visualised in Figure 1.

The multi-layer perceptron was centric to this project due to the ability to train its weights on some data to approximate some function, with the data being embeddings (discussed on page 15) of music notes and the function being one that predicts the next note in a sequence.

A *feed-forward neural network* is a type of neural network in which a neuron's activation signals always go forward to the next layer; in terms of graph theory, the network is acyclic. An illustration is given in Figure 1.

Backpropagation is an algorithm which is used in *supervised learning* (training a machine to give a certain output for a certain input with a set of examples) to train a feed-forward neural network by finding the derivative of the *loss function* with respect to the network parameters. It is often used in *gradient descent* – an algorithm which optimises by finding minimum turning points of a function – due to its gradient finding abilities.

5.2.3 Recurrent Neural Network

A *recurrent neural network*, visualised in Figure 4, extends neural networks into the time dimension by allowing for activation signals of neurons to be fed back into the network during the next input, essentially giving the network a memory of previous inputs. This gives rise to an ability to model sequences of inputs, since the network's memory of previous inputs can be used in conjunction with the current input where each input is a member of the sequence.

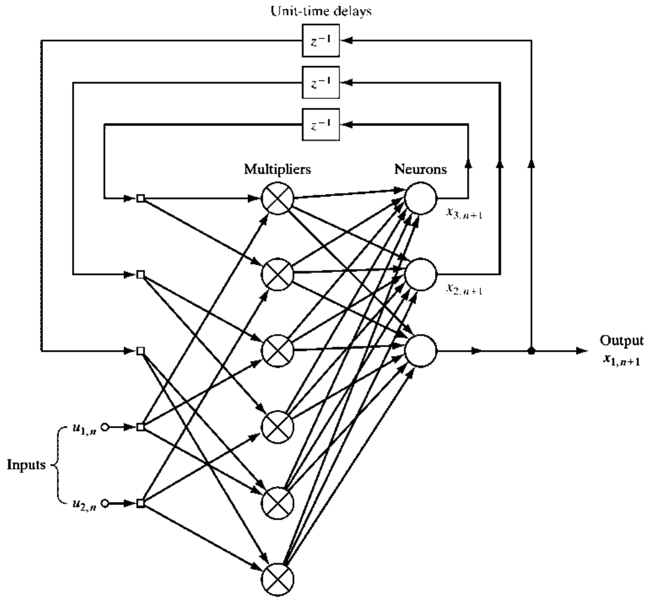


FIGURE 4 – RECURRENT NEURAL NETWORKVWITH TWO INPUTS AND ONE OUTPUT (HAYKIN, 2009)

An immediate issue which arises here however is that the memory created by this is extremely short term and is only useful for short sequences of inputs. This is known as the *vanishing gradient* problem (Hochreiter, 1998). A somewhat effective solution to this is to use a *Long Short-Term Memory* (*LSTM*) network which extends a recurrent neural network to include a memory cell which encodes past inputs and improves retention time. This cell is gated, allowing the flow of information from the cell to be treated as a parameter (Hochreiter & Schmidhuber, 1997).

The ability of an LSTM recurrent neural network to capture patterns in sequences was useful in the case of this project since monophonic music has a sequential nature, with each note being an event which could be treated as an input.

5.2.4 Input Representations

The inputs of a neural network are usually multi-dimensional vectors which are used to represent some input data. Two options worth mentioning are *one-hot vectors* and *dense embedding vectors*.

One-hot vectors are a multi-dimensional, *sparse representation* of symbols in a language, where the number of dimensions is the number of symbols in the language. By enumerating the set of symbols, each symbol can be represented by a vector with a one in the dimension of the symbol's enumeration number and zeroes in all other dimensions. The advantage to this is that it is very easy to enumerate a set of symbols however this representation fails to capture any relationships between symbols since all vectors are the same distance apart from each other. A neural network predicting the next move made in a game of rock, paper, scissors might use the following enumeration to form a one-hot encoding used for input:

$$Enum = \{1{:}\,Rock, 2{:}\,Paper, 3{:}\,Scissors\}$$

$$Rock = \begin{matrix}1\\0\\0\end{matrix}, \quad Paper = \begin{matrix}0\\1\\0\end{matrix}, \quad Scissors = \begin{matrix}0\\0\\1\end{matrix}$$

It might be worth noting that it is not useful to represent a set of symbols simply by their enumerations in a single dimension because this implies that two symbols with numbers which are similar have similar features. For example, a representation which directly uses the enumeration of the English alphabet $a = 1,\ b = 2,\ c = 3, \ldots$ means that a is closer to b than it is to c which isn't generally a valuable relationship.

Dense embedding vectors, also known at *feature embeddings*, are a multi-dimensional representation of symbols where each dimension corresponds to some feature in the language (Goldberg, 2017). This allows for relationships between symbols to be encoded by the distance between their vectors.

Embeddings are often initialised randomly (Kocmi & Bojar, 2017) or pre-trained embeddings are used, commonly being trained by Word2vec (Mikolov, et al., 2013) or GloVe (Pennington, et al., 2014); Word2Vec and GloVe are models that take a set of training data and return a set of word embeddings in a high-dimensional vector space, with the distance between the embeddings of any two words being determined by how often those words can be found in the same context in the training data.

"You shall know a word by the company it keeps."

-John R. Firth (1957).

While pre-trained embeddings give significant benefits, such as shorter training time and reducing the risk of over-fitting (Trevett, et al., 2018), it should be noted that the language feature of a dimension when using trained embeddings is not always intuitive or even understandable by humans.

Since no pre-trained embeddings were applicable to the proposed method of music generation in this project, embeddings will be devised for the tokens used to represent the music notes and symbols in the training data.

5.2.5 Multi-Task Learning

Generally, in machine learning, training a neural network will focus on one specific target; a text prediction model which simply predicts the next character in a sequence. *Multi-task learning* (*MTL*) is an approach which broadens the goals when training a neural network to include multiple related outputs.

MTL utilises a *main task* and one or more related *auxiliary tasks* with a loss function for each task. Minimising all of these losses together has the effect of improving the results of the main task (Ruder, 2017).

Two approaches to MTL include *soft parameter sharing* and *hard parameter sharing*. These determine how the hidden layers are connected for separate tasks. When using hard parameter sharing – an example of which is shown in Figure 5, hidden layers are shared for all tasks, which has the effect of reducing the likelihood of overfitting occurring. Alternatively, soft parameter sharing – demonstrated in Figure 6 – could be used, which uses a separate network for each task, then shares the hidden layers between the networks.

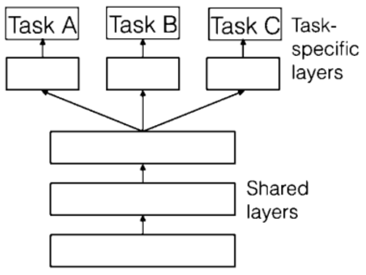


FIGURE 5 – HARD PARAMETER SHARING (RUDER, 2017)

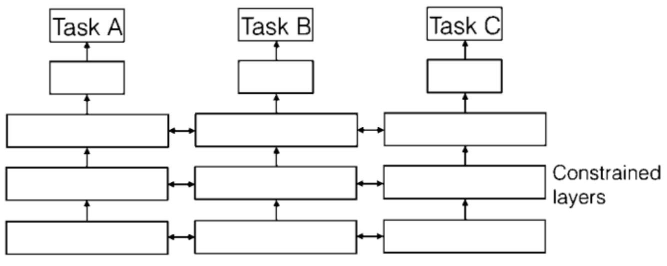


FIGURE 6 – SOFT PARAMETER SHARING (RUDER, 2017)

This project demonstrated the use of multi-task learning on music by treating note prediction as the main task and following structure and harmony as auxiliary tasks.

5.2.6 Hierarchical Neural Networks

Hierarchical neural networks can be used to take advantage of natural hierarchical structures found in language, for example the descending hierarchical *levels* shown in Figure 7. The idea is to capture and use semantic information from multiple levels to improve the model's overall "awareness" of the context of elements at each level (Sordoni, et al., 2015).

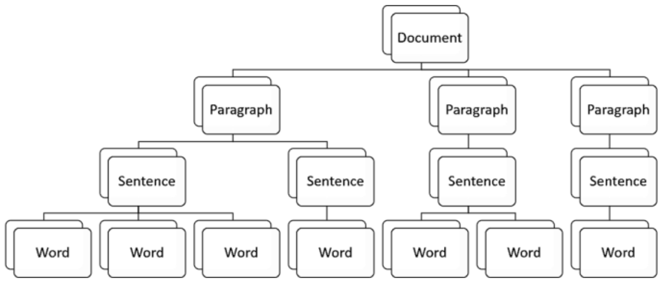


FIGURE 7 – HIERARCHY OF A TEXTUAL DOCUMENT

This technique is a solution to the vanishing gradient problem in the sense that given a textual paragraph discussing some concept, where a standard LSTM network would put less emphasis on tokens earlier in a sentence and likely stray from the initial topic, a hierarchical network could have a level devoted to a representation of the concept (in the paragraph) which is used for the prediction of the next token at the word level, hence the ability to retain the topic.

The hierarchical structure of music can be given like in Figure 8; this can be used as a guide as to which levels to use in the model. This project aimed to involve the creation of a hierarchical neural network to predict the next music note in a sequence, encoding information at the bar level and possibly song level to assist in the prediction of notes at the note level.

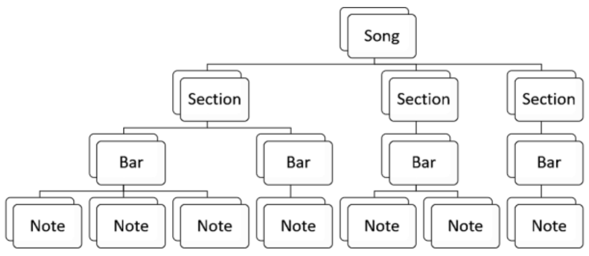


FIGURE 8 – HIERARCHY OF A SONG

5.3 Music Theory

Having at least an elementary understanding of music theory will be useful to follow the reasoning of some of the design choices in this project.

A *music score* is a visual medium used to represent musical sounds. It can be read similarly to writing, containing symbols placed on a *staff* that reads from left to right then top to bottom.

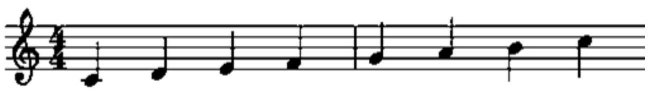


FIGURE 9 – ASCENDING C MAJOR SCALE

A *music note* is a symbol on a score which represents a sound. Each music note has various properties that can be notated which describe features such as *pitch*, *rhythm*, accent and so on.

A *rest* (Figure 11) is a symbol on a music score which represents silence. The only properties that a rest may have are rhythmic.

5.3.1 Rhythm

Rhythm describes the positioning of music notes over time. Due to the lengthwise scoring of rhythm – similar to time often being the x axis on graphs – it is said to be the horizontal aspect of music.

A notable property of music notes is *duration*, with each note having a length that is divisible by the *time signature*. A time signature is a pair of integers $\frac{x}{y}$ where x denotes the number of *beats* in a *bar* and y denotes the duration of those beats, where when $y = 1$, a beat is a 𝅝 (whole note). Figure 10 shows how the whole note relates to divisions of itself.

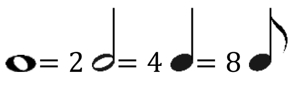


FIGURE 10 – RELATIVE DURATIONS OF MUSIC NOTES

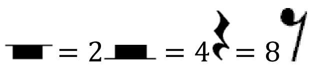


FIGURE 11 – RELATIVE DURATIONS OF RESTS

A note can also be dotted to multiply its duration by 1.5.

Rhythm is important to understand in this project since a music generating algorithm must be able to follow the rules of how many beats are in a bar of music, given a time signature.

5.3.2 Melody

Melody is the use of pitch and rhythm in a sequential manner to tell a musical story; at core level, it is a sequence of musical notes and rests, with only one pitch sounding at once. An example of a simple melody is given in Figure 12.

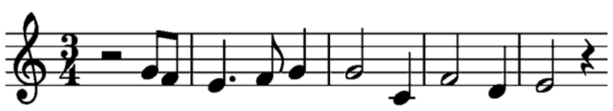


FIGURE 12 – MELODY IN C MAJOR, 3/4 TIME

Monophony refers to music which has a single melody; this project concerned itself with the melodies found in monophonic Irish folk music.

5.3.3 Harmony

The pitch of a music note determines the acoustic frequency of a music note. The difference between the pitches of two music notes is called an *interval*, the smallest of which is called a *semitone*. For two notes where the first note is half the frequency of the second, the interval is called an *octave*. In Western harmony, there are twelve semitones in an octave.

When three or more music notes with different pitches are sounded simultaneously, the result is said to be a *chord*, an example of which is notated in Figure 13. Chords are often used as pivots in music, with the pitches of music notes in melodies tending to focus around the core pitches of a chord.

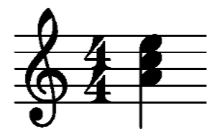


FIGURE 13 – AN A MINOR CHORD

Harmony, in essence, describes the existence of relationships between the pitches of music notes; occurring when multiple notes with different pitches are played simultaneously – harmony can also be implied by a melody when it outlines a chord, for example an arpeggio, which involves playing all the notes of a chord in succession rather than simultaneously. It is said to be the vertical aspect of music due to the fact that pitches are scored on the vertical axis in sheet music, as shown by the ascending C major scale in Figure 9.

Harmony could have been used for an auxiliary task of multitask learning in this project, with the task being devoted to predicting chord changes in a melody.

5.3.4 Structure

Music is generally a highly structured art form, having repetition on small scale (beats and bars) and large scale (*sections*). Irish folk music – the style which this project focuses on – typically uses *binary form*, having A and B sections (usually of identical length – referred to as symmetrical) which are played in an AB or ABA fashion. The music shown in Figure 14 utilises symmetrical binary form with the first eight bars being section A followed by the last eight bars which is section B.

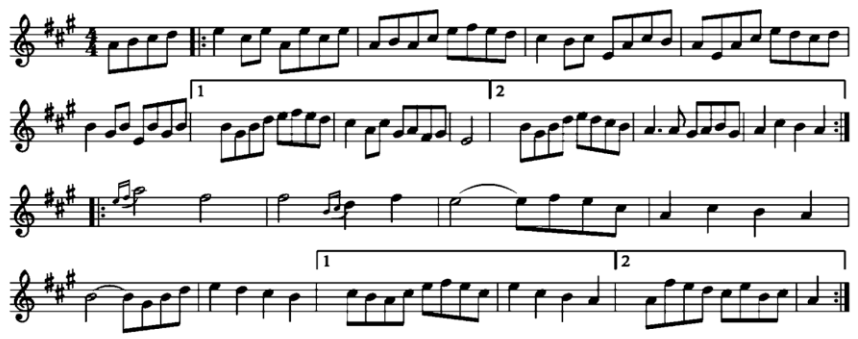


FIGURE 14 – SHEET MUSIC OF "HOMAGE A EDMOND PARIZEAU", AN IRISH REEL (COWAN, 2018)

Structure was used as an auxiliary task for multitask learning, involving deciding on when a new section starts and ideally guiding the main task to create repetition. It would also have been highly suitable for modelling within a hierarchical neural network since the network could capture the multiple scales of structure mentioned above.

6 Related Work

6.1 Music Generation

Using deep learning to generate music is far from a brand-new concept; a number of papers have covered many aspects in the field.

An RNN using LSTM was built by Sturm et al. and trained on a dataset of thousands of Irish folk tunes (Sturm, et al., 2016); the dataset will be borrowed for this project and lessons will be taken due to the similarity between projects. To generate pieces of music, a musical "idea" was given to the network as a seed and proceeding notes were predicted. The resulting compositions generally followed the rules of transcription and statistically followed the binary form structure common to the genre; glancing at individual pieces also showed use of finer structural details such as repetition.

On the other hand, this model appeared not to effectively model harmony, with the implied harmony outlined by the melody being described somewhat "aimless" and "awkward". These problems would likely be remedied by adding harmony as an auxiliary task in a multi-tasking architecture, or, since the harmony of folk music is relatively dependent on structure, using structure as an auxiliary task.

An important note covered by the paper is whether the model is simply copying parts of melodies from the training data: when inspected, some parts are found to be copied, sometimes used in different contexts original to the generated music, and some ideas are entirely unique.

The above dataset was also used in a paper exploring the best parameters to be used with the goal of comparing generated music with music composed by a human. (Cowan, 2018) Using a recurrent neural network and giving prompts of various bar lengths, the conclusion was drawn that the model could effectively reproduce the style and gave a surprising result of human participants preferring the generated music; especially interesting since over three quarters of participants were musicians. Unfortunately, the sample size of participants for this study was extremely small so results cannot be taken conclusively.

Another paper by Sears et al. explores music generation and prediction with an emphasis on harmony, using the music of numerous classical composers covering chorales, quartets, symphonies and piano pieces (Sears, et al., 2018). The project makes no effort to explore the structure of the music, focusing instead on breaking down the chords found at every point of time in the music.

Chords were encoded by regarding the bass note and noting intervals above it within an octave – the intervals being represented as permutations (ignoring the order) to reduce input space. The embeddings for the chords were a learned vector representation.

An RNN using LSTM was used and contrasted with other natural language processing models to find LSTM being the most successful RNN but less successful against Finite Context models. This is an interesting result demonstrating the limitation of RNNs, however having an especially large vocabulary warrants an equally large dataset; the dataset used in this project was both inconsistent – using both Scott Joplin's ragtime piano and Beethoven's symphonies – and small – consisting of only a thousand compositions.

6.2 Multi-Task Learning

While there are no examples of multi-task learning being applied in a musical context, the alignment between music and natural language was used to gather and discuss useful related work.

Multi-task learning can be found in some recent language prediction models, including that of the LSTM network MKAL (Sun, et al., 2016). The paper proposes a multi-task LSTM model with a relation classification – the classification of relationships between two words or terms in a sentence – task as an auxiliary task to the main task of inferring non-explicit facts from textual data.

The paper uses the SNLI dataset (Bowman, et al., 2015) – a dataset containing triples of a statement, a hypothesis relating to the statement and if the hypothesis entails, contradicts or does neither to the statement – to express the excellent improvement of performance by the addition of the auxiliary task to the model; prior to multi-task learning, it gives an accuracy of 78.8% whereas with the auxiliary task, it performs with 82.5%. This improvement, when compared to other architecture changes made to the model, was the most significant, thus suggesting the usefulness of applying multi-task learning to text classification.

Task supervision (providing input, output pairs for training) is usually always done at the final layer in a network, however it is possible, and indeed arguably better, to place supervised tasks at lower levels in the network (Yaov & Søgaard, 2016). The paper tests benchmark language modelling tasks – part of speech tagging, syntactic chunking, and CCG supertagging – on multi-task RNNs in an experiment to show the effect of placing a task at different layers above or below another task.

The paper concludes that if there exists a hierarchy between the tasks at hand, placing a task's output at a lower layer in the network is a useful modelling technique. Thinking of the hierarchy of music could have been useful when considering specific design for the multi-task learning network for music generation.

6.3 Hierarchical Neural Networks

Similar to multi-task learning, no directly related work which explores the application of hierarchical neural network models to music is available to discuss, however, again, it is fairly commonplace for hierarchical neural networks to be used in natural language processing.

In a paper by Sordoni et al., the hierarchical structure of language is considered for the modelling of a hierarchical recurrent neural network to encode and decode predictions for search queries (Sordoni, et al., 2015). The model utilises two levels: the session level and the query level. A query is considered to be a sequence of words which make up a search term, encoded as a single vector; a session is the sequence of queries that a person searches in a short space of time in an attempt to reach some goal information. The session level aims to encode information which allows the query level to predict the words in the next query in the sequence. A visualisation of this idea is seen in Figure 15.

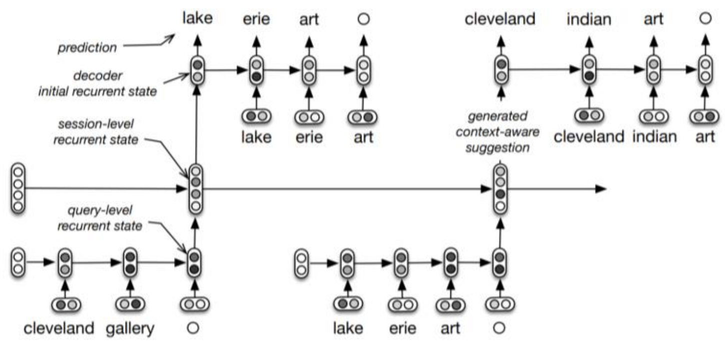


FIGURE 15 – HIERARCHICAL RECURRENT ENCODER-DECODER FOR QUERY PREDICTION (SORDONI, ET AL., 2015)

Aside from positive methodical testing, the paper studied the subjective usefulness of the generated queries during a user study and found that the hierarchical element gave a superior performance, creating a higher proportion of "useful" and "somewhat useful" queries to standard techniques.

The paper concludes by talking about the possible future application of language modelling and other problems within the field of NLP which gave optimistic signals for the project here.

A hierarchical neural model was applied to human dialogue to predict a response given some words (Serban, et al., 2015). It uses three levels of hierarchy: the word level – which are encoded using trained dense embedding vectors, the utterance level – sequences of words as a hidden state, and the dialogue (or context) level – a hidden state memory of the encodings of all previous utterances. The network used was a typical RNN extended like the above network (Sordoni, et al., 2015), considering instead that a dialogue consists of a sequence of utterances, an utterance being a sequence of words.

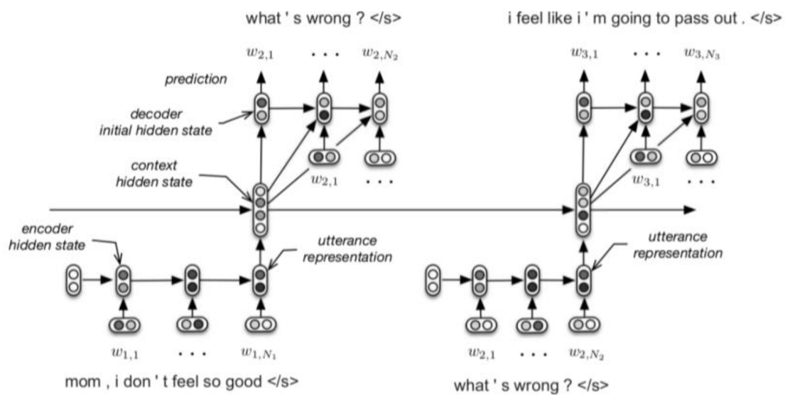

what ' s wrong ? </s>
i feel like i ' m going to pass out . </s>
$w_{2,1}$ . . . w_{2,N_2}
$w_{3,1}$ . . . w_{3,N_3}
prediction
decoder initial hidden state
context hidden state
$w_{2,1}$ . . .
$w_{3,1}$ . . .
encoder hidden state
utterance representation
utterance representation
$w_{1,1}$ . . . w_{1,N_1}
mom , i don ' t feel so good </s>
$w_{2,1}$ . . . w_{2,N_2}
what ' s wrong ? </s>

FIGURE 16 – HIERARCHICAL RECURRENT ENCODER-DECODER FOR PREDICTING THE NEXT UTTERANCE IN A THREE TURN DIALOGUE (SERBAN, ET AL., 2015)

Predictions are made at the level of words and utterances, with the network predicting the next utterance vector based on the context state and the next word using previous words and the utterance vector. Utterances are always terminated with an end symbol; upon prediction of this symbol, prediction at the word level ceases. The ideas demonstrated in this paper can be thought of generally to form the core of using hierarchical neural networks for natural language prediction, which would have created extremely useful groundwork for the project of music generation.

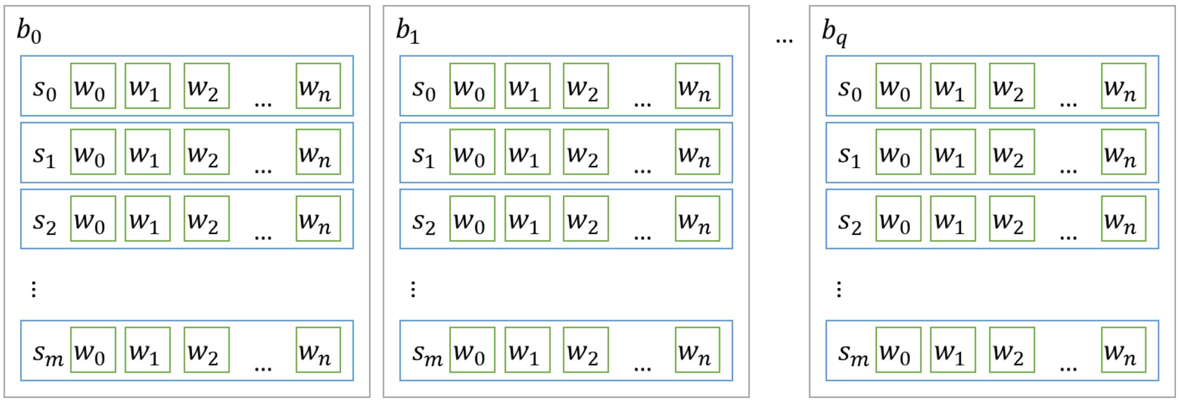
Another relevant work gives a syntax which formalises chord sequences and cadences found in Jazz music (Granroth-Wilding & Steedman, 2014). It considers the structural significance of cadences – harmonic “conclusions” of sections of music using chord sequences – building trees to capture their function in the music. The paper includes an excellent, in-depth discussion on musical harmony, necessary for building the grammar it proposes. The ideas there indicate that it is not presumptuous to relate the hierarchical structures found in music to those found in natural languages; a useful statement because those found in natural languages are commonly modelled in the designs of hierarchical neural networks.

7 System Development

7.1 PyTorch

PyTorch is a library for Python which provides methods for constructing neural networks and handling data in the form of tensors. Particularly it condenses building and experimenting with LSTM RNNs into a collection of modules which can be fitted together in any required manner, making it suitable for this project.

PyTorch introduces the Tensor class with the purpose of having multi-dimensional vectors designed for deep learning functionality. Dimensionality was important to consider here; for NLP, datasets are typically three-dimensional Tensors, with the first dimension being the batches, the second being the sentences (in this case, tunes) and third being a numerical value corresponding to the symbol – a visualisation of this can be found in Figure 17.



b_0
b_1
...
b_q
s_0 w_0 w_1 w_2 ... w_n
s_1 w_0 w_1 w_2 ... w_n
s_2 w_0 w_1 w_2 ... w_n
⋮
s_m w_0 w_1 w_2 ... w_n

FIGURE 17 – VISUALISATION OF TRAINING DATA DIMENSIONALITY (b – BATCH, s – SENTENCE, w – WORD)

7.2 Network Design

Two main model architectures were created: a baseline LSTM RNN and a Multi-task LSTM RNN with a main task and auxiliary task. Both models had modifiable hyperparameters to allow for experimentation. Network parameters were randomised upon initialisation.

The Multitask network extended that of the baseline RNN by including decoding layers for tag symbols which received their input from the LSTM layer. A high-level comparison between the two architectures which were implemented is displayed in Figure 18.

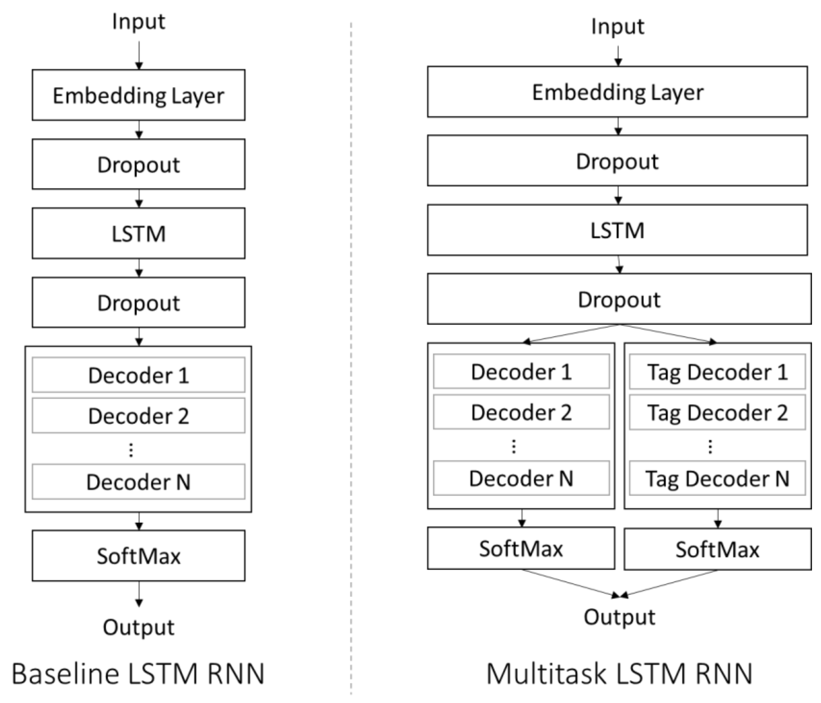


FIGURE 18 – ARCHITECTURE COMPARISON BETWEEN BASELINE RNN AND MULTITASK RNN (EACH WITH N HIDDEN LAYERS)

Using the example of abc music notation given in Figure 19, a brief design of the models this project aimed to explore are visualised below. w_n will be used to refer to a symbol in the string of music notes starting from $w_0 =$ |:, $w_1 = A$ and so on.

```
X: 1
T: The Hag at The Churn
R: jig
M: 6/8
L: 1/8
K: Dmix
|: A2G ADD|A2G Adc|A2G ADD|EFG EFG :|
|AdB c3|Add efg|AdB c2A|GEG AED|
AdB c3|Add efg|age dcA|GEG AED||
```

FIGURE 19 – AN IRISH JIG[2] IN ABC NOTATION

7.2.1 MULTI-TASK LEARNING

The text in Figure 20 is the excerpt from Figure 19 with added structure and harmony information at each symbol. Line one uses the idea of the sections in binary form but allowing up to six different sections, using a label from A through E to represent what section the corresponding note on line three belongs to. Line two denotes the chord to be played during the corresponding note on line three. Line three is the original abc notation of the piece. Spaces are included in lines one and two where abc symbols are multiple characters long.

```
Line 1: A  AA  AAAAAAA AAAAAAA AAAAAAA AAAAA  BBBB BBBBBB BBBBBBB BBBBBBB BBBB
Line 2: D  DD  DDDDDDD DDDDDDD DDDDCCC CCC C  DDDD DDDDDD CCCCDDD DDDDCCC DDDD
Line 3: |: A2G ADD|A2G Adc|A2G ADD|EFG EFG :| |AdB c3|Add efg|AdB c2A|GEG AED|


Line 1: BBB BBBBBB BBBBBBB BBBBBBB BBBB
Line 2: DDD DDDDDD CCCCAAA DDDDCCC DDDD
Line 3: AdB c3|Add efg|age dcA|GEG AED||
```

FIGURE 20 – THE ABOVE IRISH JIG WITH SECTION AND HARMONY INFORMATION EXTRACTED

Figure 21 illustrates the flow of inputs and outputs in the multi-task learning model. The main task of the network was to predict the next note in the sequence (line three) given all previous symbols on line three. The auxiliary tasks were to predict the symbols on lines one and two given the current and previous symbols on line three. A symbol n on a specific line L is given $w_{L,n}$ (demonstrated with annotations in the figure). The

[2] Source: https://thesession.org/tunes/829

state of the network at a timestep t is shown as S_t. A predicted probability distribution is given as $\hat{w}$ and decoder output is given $\tilde{w}$.

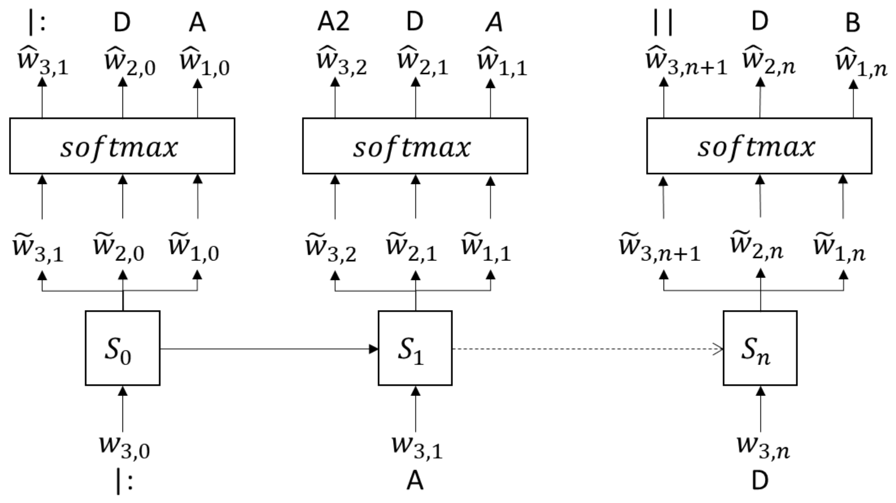

|: D A
$\hat{w}_{3,1}$ $\hat{w}_{2,0}$ $\hat{w}_{1,0}$
A2 D A
$\hat{w}_{3,2}$ $\hat{w}_{2,1}$ $\hat{w}_{1,1}$
|| D B
$\hat{w}_{3,n+1}$ $\hat{w}_{2,n}$ $\hat{w}_{1,n}$
$softmax$
$softmax$
$softmax$
$\tilde{w}_{3,1}$ $\tilde{w}_{2,0}$ $\tilde{w}_{1,0}$
$\tilde{w}_{3,2}$ $\tilde{w}_{2,1}$ $\tilde{w}_{1,1}$
$\tilde{w}_{3,n+1}$ $\tilde{w}_{2,n}$ $\tilde{w}_{1,n}$
S_0
S_1
S_n
$w_{3,0}$ |:
$w_{3,1}$ A
$w_{3,n}$ D

FIGURE 21 – THE EXAMPLE IRISH JIG AS PROCESSED BY A MULTI-TASK LEARNING MODEL

This design relied on assuming chord and structure information can be easily and effectively extracted from Irish folk music since the dataset will contain only melody information. This assumption was grounded on the objective simplicity of the harmony of folk music, so by using the pitches of music notes present in a melody, chords from the key of a piece can be chosen algorithmically. This proved not to be the case as discussed in section 0. Additionally, the assumption that the structure of the genre is simple to extract was based on general use of symmetric binary form in these pieces; again, labelling of the structure can be done algorithmically. The method of extraction of these is described in further detail in section 0.

7.2.2 Hierarchical Neural Network

The diagram in Figure 22 depicts a hierarchical neural network with three levels: note, bar, and section. Each state of the network is shown $S_{L,n}$ with index n of the states at level L. $\mathbf{0}$ represents the zero vector.

At the note level, each note would be used to predict the next symbol in the sequence, like a normal RNN, however, it would also use the bar state vector as an input for prediction.

At the start of every bar, the vectors of all the notes in the previous bar would be compressed together (possibly summed) and used as input for predicting the vector at the next bar state, alongside inputs: the section state vector and the previous bar state vector.

At the start of every section, the previous section state vector, and the compressed bar state vectors (again summed) would be used as inputs to predict the vector at the next section state.

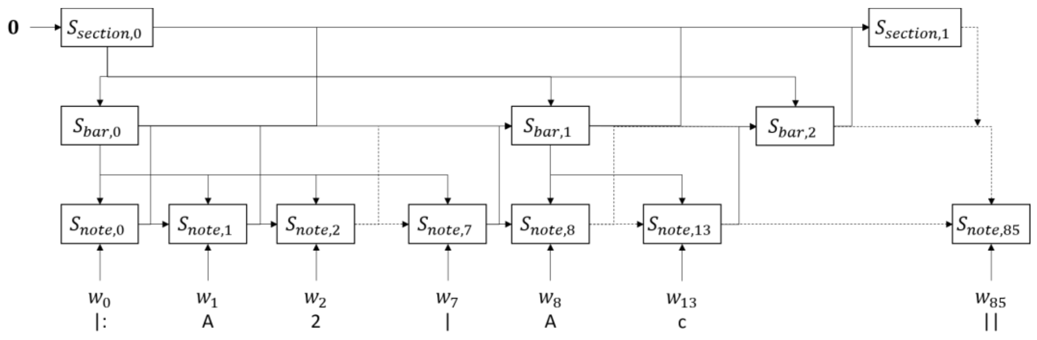


FIGURE 22 – THE EXAMPLE IRISH JIG AS PROCESSED BY A HIERARCHICAL NEURAL NETWORK

Upon generation/encountering of an end of bar symbol – **|**, the network would move to the next bar, therefore allowing variable bar length depending on the note level generation. Additionally, the generation of an end symbol </S> could cause generation to cease.

A potential problem arose while considering when to start a new section, a possible solution being to use the section data proposed in Figure 20 from Multi-Task Learning and use a dedicated new section symbol; this would be a special symbol which causes a new section to be started.

Note that for generation, section 0 could be fed with a random vector as a seed. A likely more effective approach, however, would be to calculate a section vector for a section of music in the training data and generate based on that.

A possible addition to this design worth considering would be a level which encodes the key, time signature, timing division and possibly title of the piece. This would have the benefit of ensuring avoidance of the music straying from the original key and failing to generate the correct number of notes in a bar. This information might be most useful being fed straight to the note level rather than going through section and bar.

This design relied on the aforementioned assumption of straightforward extraction of structure information from music in the genre.

Due to the complexity of building a complete grammar for abc notation or extracting hierarchical information from the dataset with some other method, especially alongside the technical difficulties associated with deciding when to start new sections or bars during generation, it was decided not to implement a hierarchical neural network and instead focus resources on optimising Multitask learning to as full an effect as possible.

7.3 Training

To train the models, a dataset[3] consisting of above 46,000 pieces of Irish folk music in abc notation was used. The pieces were parsed to retrieve a set of tokens, for which enumerations were created, and a set of pairs of sequences of notes and next notes of those sequences. PyTorch provided functionality for training networks, in this case by calculating loss and using stochastic gradient descent to optimise the network parameters.

Upon running the training script, the dataset was loaded and converted to a Tensor with values corresponding to the one-hot vector representations for the symbols, which were derived from the enumerations. This data was shuffled then organised into a number of batches – each containing 32 sentences. Batching is a feature of PyTorch which allows training to be sped up by performing many loss calculations in parallel on a GPU.

7.3.1 Loss

Models were saved automatically during training for the sake of allowing the training to be cancelled at any stage while retaining the best model. The smallest loss was used to make comparisons and save optimal network parameters – it was saved alongside the model to allow continuation of training.

For calculating the loss of the Multi-task model, the total loss was calculated by adding the loss of the main task to that of the auxiliary task multiplied by the loss of a hyperparameter λ.

$$L = L_{main} + \lambda L_{aux}$$

[3] The data can be found at: https://github.com/IraKorshunova/folk-rnn/tree/master/data

7.3.2 Learning Rate

The learning rate was dynamic, starting at an initial value and decreasing exponentially with the number of epochs. Upon each epoch without any saved models, the learning rate "creeps" up in an attempt to escape a high-dimensional trough of the function – when the model is saved again, it returns to its original decreased rate at the beginning of the next epoch.

7.3.3 Datasets

A dataset manager class was designed to cope with the data for a number of reasons, including inconsistency with order of enumeration and the slowness of processing the entire text file each time training, evaluation or generating was performed. An important benefit that arose from this design was the ability to partition data into subsets for training (88% of the original dataset), validation (10%) – used for calculating perplexity, and testing (2%) – which would have been used to gather prompts for generation.

Each sentence in the set had <S> inserted at the beginning and </S>, corresponding to start and end symbols. Due to the requirement that all sentences in batched data must be the same length, functionality was given to the dataset manager to pad shorter sentences so that all sentences were as long as the longest sentence in the dataset. A pad symbol <P> was used to represent padding and was ignored when calculating loss.

An issue which arose from this design was the refactoring which had to take place when the dataset was revised, however the loss here was not quality of results but of code legibility.

7.4 Data Tagging

An algorithm to tag music from the dataset with section labels was devised, using both the syntax of repeats in abc notation alongside counting the number of bars to extract sections of the piece and assign a label of "A", "B", "C", "D", or "E". The number of sections was limited between two and five – for example, a piece with one long A section was not allowed.

The described algorithm could not be fitted to 18.94% of the original dataset. Failures included reasons such as the piece was returned as one long A section or had sections beyond the "E" section – these were not considered useful, since they did not provide relevant structural data for the task and would likely confuse the model, and were discarded from the dataset.

An initial goal was to extract chord information from the dataset by using the melody to imply a chord at certain time intervals; however, a number of issues were encountered in attempting to do this. The tunes often contained sharps and flats beyond the initial key signature of the piece, which added a level of complexity to figuring out the implied chord. This was doable, however it increased the size of the vocabulary of chord symbols by a large amount, since all chromatic major and minor chords had to be accounted for, risking causing some confusion to the network, particularly if the wrong chord was derived in even just a few cases. An outcome of this is that testing would have needed to be extensive and likely unfeasible for this project.

Another issue with chord information is that it is extremely difficult to tell, algorithmically, where a chord change should happen; this is completely intuitive for a musician but modelling this would likely be an entire project alone. It was concluded that chord information should be omitted.

7.5 HYPERPARAMETER EXPERIMENTATION

A number of experiments were created with various different values for each hyperparameter and assigning each experiment a number. Each experiment was trained for approximately fifty epochs, saved, and evaluated using perplexity.

The list of hyperparameters considered follows:

- Hidden layer size
- Embedding size
- Embeddings dropout
- LSTM dropout
- Number of hidden layers
- Auxiliary Task Weight λ (Multitask RNN only)

7.5.1 PERPLEXITY

Perplexity is a statistical term denoting effectiveness of a probability distribution at predicting some data; it can be described as a model's uncertainty at making predictions. Perplexity was useful for evaluating and comparing the models mathematically prior to testing with humans.

The perplexity PP of a sentence W containing words $w_1 \dots w_n$ was calculated by summing log probabilities as follows: $PP(W) = e^{-l}$ where $l = \frac{\sum_{i=1}^{n} \log(P(w_i))}{n}$.

Each experimental model was tested by using it to make predictions on the training dataset. If the distribution of the predictions accurately reflects the actual next note in the sequence, the perplexity will be lower, and the model will be considered more ideal. Perplexities were recorded for each model with successful models being used to generate music for the human evaluation stage.

7.6 Generation

To generate the music, two approaches were used for decoding output from the model: greedy search and random sampling. The model was given a prompt string containing only a start symbol and a loop was run by feeding the network the entire previous string of symbols and using one of the decoding methods to select a symbol from the resulting distribution at each step, continuing to iterate until either the maximum sentence length was reached or the selected symbol was an end symbol.

The decision was made to treat time and key signatures as symbols to allow the model more initial freedom when expressing ideas as well as leaving open the option of giving a prompt with particular signatures. The model (almost always) predicted a time signature then key signature immediately after the start symbol before predicting any notes since in the training data, there were no circumstances in which the start symbol was not followed by a time signature then a key signature.

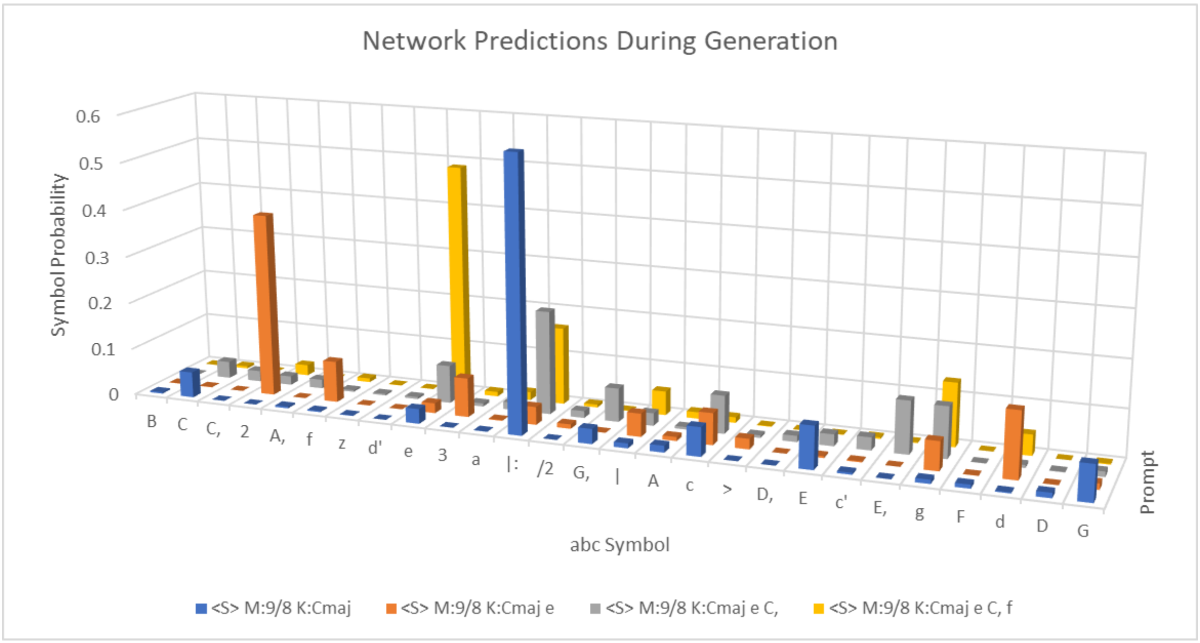


FIGURE 23 – EXAMPLES OF DISTRIBUTIONS PRODUCED BY THE NETWORK (CONSISTENTLY UNLIKELY SYMBOLS WERE FILTERED OUT) WITH EACH COLOURED SERIES CORRESPONDING TO THE DISTRIBUTION GENERATED BY A SINGLE PROMPT

7.6.1 Greedy Search

An implementation of Greedy search was used to always select the symbol that the network predicted with the highest confidence by simply taking the $argmax$ of the output distribution – the symbol corresponding to the highest spike for each prompt in Figure 23, e.g. greedy search given the first prompt would give |:. The tunes generated by this approach were extremely repetitive to the point of repeating one or two note phrases for many bars as seen in Figure 24.

This is probably because certain notes like the tonic – C – or the dominant – G – were encountered very often during the training data so to reduce loss, the network could simply give out a high probability for the most common notes, no matter what notes came before.

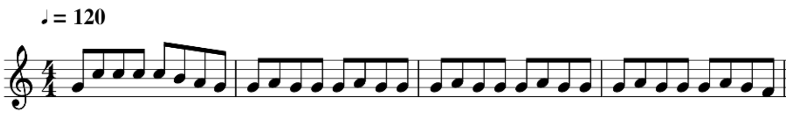


FIGURE 24 – EXCERPT OF TUNE GENERATED BY GREEDY SEARCH

7.6.2 Random Sampling

The output from the network was a probability distribution due to the network applying $softmax$ before output, therefore an algorithm could be used to pick a random element from the list of symbols corresponding to the distribution, with weighting towards symbols with higher probabilities; the *choice* function from *NumPy* library provides this functionality. For a prompt in Figure 23, symbols corresponding with higher spikes would be more likely to be chosen but there existed opportunity for those with smaller spikes to be selected, for e.g. for the second prompt, 2 is most likely to be chosen but F, 3 or d may also be, and so on.

Tunes generated by this approach were less repetitive and on a purely subjective basis sounded more tuneful. Figure 25 shows four bars of an example of one such tune.

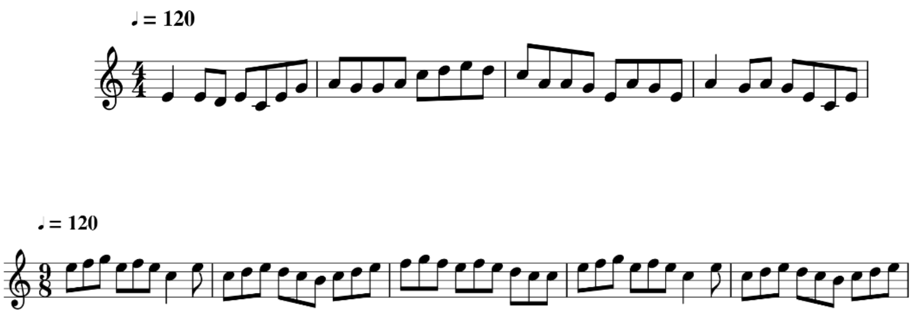


FIGURE 25 - EXCERPTS FROM TWO TUNES GENERATED BY RANDOM SAMPLING

7.6.3 ABC2MIDI

Generated music was converted into MIDI, a format allowing for audio playback using a synthesiser. abc2MIDI[4] is a software package which provides functionality to convert music written in abc notation into MIDI, allowing for audio playback. Additionally, MIDI was able to be converted into graphical scores, like the ones in Figure 25, using the command line features of MuseScore 2[5]. These capabilities were found to be essential for evaluation and discussion therefore python scripts were created to automate these upon generation.

[4] http://abc.sourceforge.net/abcMIDI/original/
[5] https://musescore.org/

8 Evaluation and Analysis

8.1 Perplexity

Model Name	Dropout	Hidden Layers	Aux-task Weight	Perplexity	Aux-task Perplexity
multitask_08	0	1	0.005	**4.253**	**2.99**
multitask_07	0	1	0.05	**4.312**	**2.074**
multitask_05	0.2	1	0.05	**4.352**	**2.126**
baseline_08	0.2	2	-	**4.417**	**-**
baseline_07	0.2	1	-	**4.814**	**-**
baseline_09	0	1	-	**4.853**	**-**

FIGURE 26 – TABLE CONTAINING NOTABLE HYPERPARAMETERS AND CALCULATED PERPLEXITIES FOR THREE BEST MODELS OF BOTH SYSTEMS (NOT ALL HYPERPARAMETERS SHOWN)

A number of experimental models were trained and evaluated using a measure of perplexity, a description of which can be found at 8.1. The three highest scoring models of each type are shown in Figure 26 – for completeness, the perplexity was also calculated for the tags generated by the auxiliary task in multi-task models.

Each experimental model was manually assigned a set of hyperparameters in attempt to find the best performing set. Once a value for single hyperparameter was found to be optimal, the majority of experiments going forward used that value.

A full table of results and hyperparameters can be found in Appendix A.

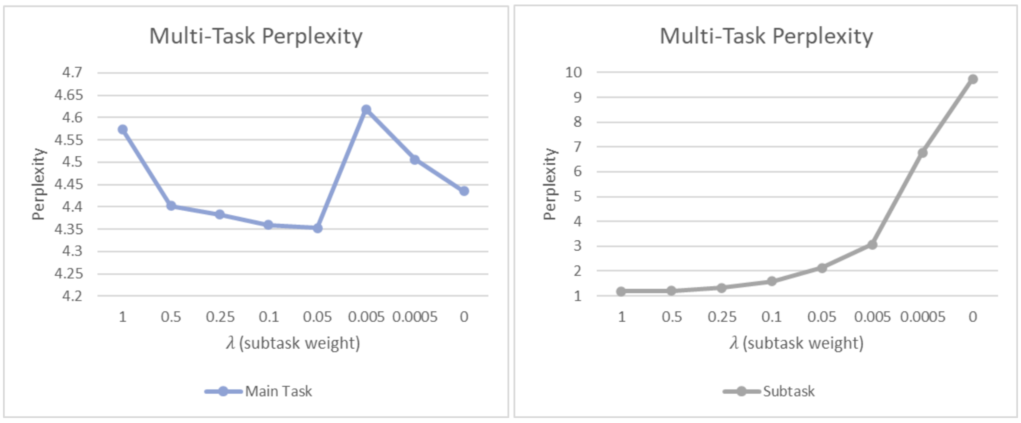


FIGURE 27 – RELATIONSHIP BETWEEN PERPLEXITY CALCULATED FOR EACH TASK AND THE WEIGHTING OF THE AUXILIARY TASK DURING LOSS CALCULATION

The first thing to point out from Figure 26 would be that the three best Multitask models perform better than the three best baseline models (this order continues in the table in Appendix A), even after extensive tuning of the various hyperparameters – in fact a comparison between a multitask model and a baseline RNN model with identical hyperparameters would consistently demonstrate improved performance; it would not be presumptuous to conclude that adding the auxiliary task of labelling AB sections in musical structure allows for better modelling of the style of music at focus.

A TREND CAN BE SEEN IN

Figure 27 between perplexity and the weighting given to the auxiliary task during calculation of loss in training which suggests the optimal weighting would be around 0.05. A large spike can be seen at 0.005, which seems somewhat unusual considering the perplexity falls again upon decreasing this value further. This may be explained by the fact that only a single model was trained for each experiment and due to the complex nature of neural networks (small changes such as the order that the dataset is presented to the network in during training or the initial starting parameters may contribute to different solutions), it would be useful to train numerous models with the same hyperparameters and find an average perplexity.

THE VALUES SHOWN IN THE GRAPH IN

Figure 27 correspond to models which used dropout; however it is interesting to note that the best performing model overall (see Figure 26) had an auxiliary task weighting of 0.005, identical to the value at the spike, but it used no dropout.

Decreasing dropout consistently lowered the perplexity of a model, seen in Figure 26 (more instances can be found in the entire table of results at Figure 38), where each model which uses dropout is worse performing than the version with the exact same hyperparameters but zero dropout.

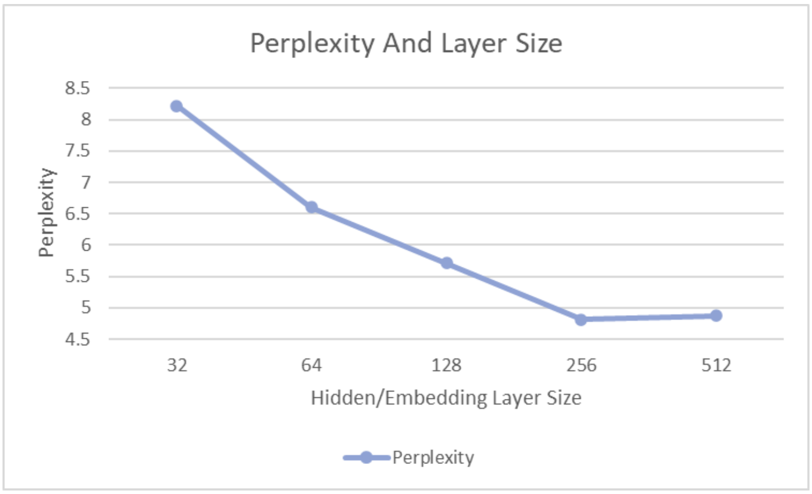


FIGURE 28 - RELATIONSHIP BETWEEN PERPLEXITY IN BASELINE MODELS AND HIDDEN/EMBEDDING LAYER SIZE

An optimal value for the layer sizes was found at 256 for both embedding and hidden layers, as seen in Figure 28. There were no further gains to be found beyond this value, in part due to the increased training and running times for larger layer sizes. The models used in this graph had common hyperparameters aside from the layer and embedding sizes.

The number of layers was kept at one after early experimentation, since any value higher gave poorer results and took far longer to train.

8.2 Human Evaluation

Since music is ultimately an art, its subjectivity made it useful to perform a human evaluation to compare the performance of the model architectures in terms of likeability and also to see if their results can be likened to that of music composed by a human.

A between-subjects experiment design was used with three separate groups of participants, each group corresponding to one of the following systems: Human, Baseline RNN and Multitask RNN. This prevented participants from being able to identify features of music from a particular system and biasing their opinion.

Before exposing participants to music, twenty tunes were generated for each RNN system. Tunes were then selected for each system by listening to and picking five which demonstrated as large a variety of musical features as possible. The generated MIDI for each selected tune was then placed into Cubase LE5, a software package for producing music, and assigned to a VST[6] track before exporting the audio to mp3 format. The chosen VST instrument emulated a flute since this instrument is prominent in performances of music in the Celtic style.

The produced tunes were placed onto a web server. The directories, file names and Mechanical Turk batch names were labelled with numbers for their corresponding system to avoid any risk of participants being exposed to a URL which gives away how the music was generated when their browser loads the music. Participants for systems 1 and 2 were not told at any point that the music they listened to had been generated by a computer and no information was shown which would suggest this.

[6] Virtual Studio Technology – used to synthesise real instruments

The Amazon Mechanical Turk platform was used to gather results from workers who would act as participants. A batch for each system was released containing five pieces each. Results from four participants for each batch were gathered.

Each individual participant was presented with five pieces of music; for each piece, they were told to listen in full and afterwards, rate their experience using a Likert scale between one and five where five was very positive and one was very negative. Each number was labelled respectively: *Hate*, *Dislike*, *Neutral*, *Like* and *Love*.

If the participant were confused, a display showing full instructions for the task could be accessed, giving an example piece of music for each number of the scale to give them an idea of how they might rate a piece. Participants were urged to wear headphones to improve quality of results.

Amazon Mechanical Turk workers were given a time limit of fifteen minutes to listen and rate a single piece of music and, upon having their submissions approved – after results were collected and analysed, a worker would be paid $1 for each tune they rated.

The interface presented to participants can be found in Appendix B.

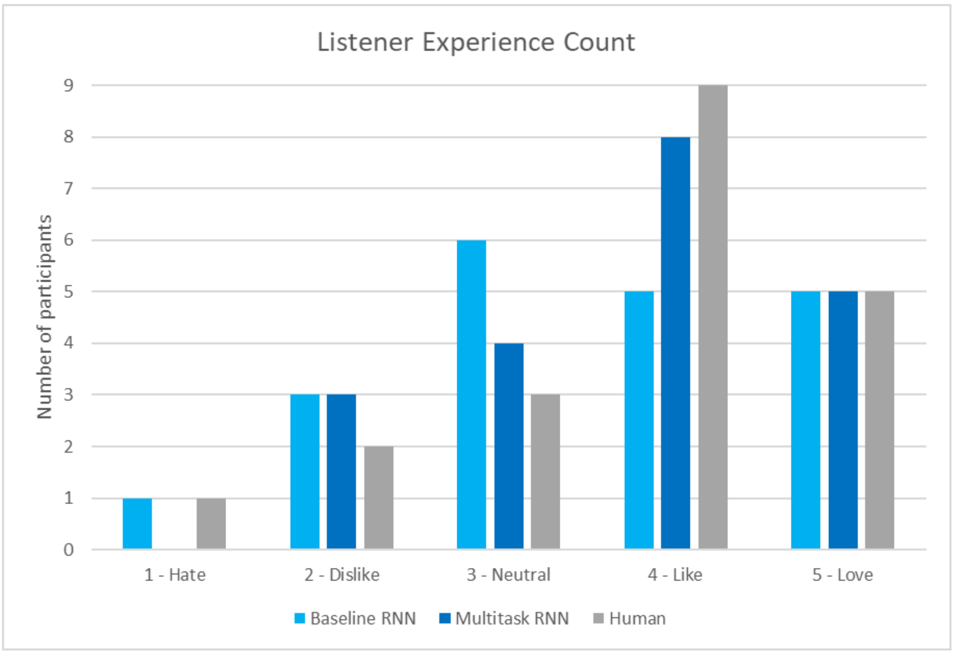


FIGURE 29 – HISTOGRAM SHOWING NUMBER OF PARTICIPANTS WHO EXPRESSED EACH SENTIMENT FOR EACH SYSTEM

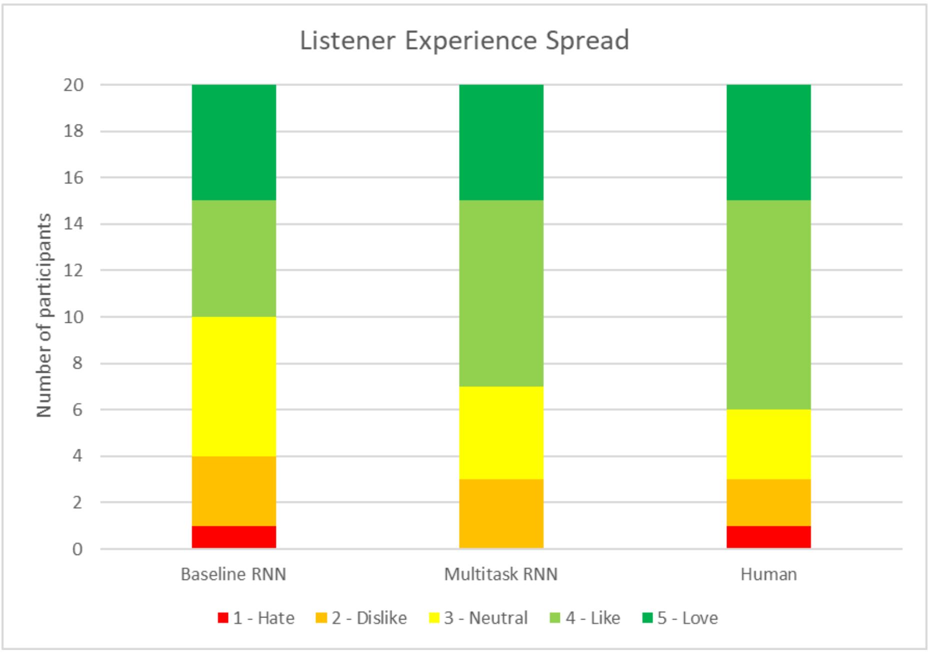


FIGURE 30 – SENTIMENT SPREAD FOR EACH SYSTEM

An initial glance at the results in Figure 29 gives the impression of a successful outcome. The number of people who *liked* the Multitask RNN music was higher than the baseline, approaching the number who *liked* the human music. Looking at Figure 30, going from left to right, the participants who chose *like* supplant those who chose *neutral* or *dislike*.

The fact that no one *hated* the music by the Multitask RNN is likely an anomaly; given more data points there would likely be a consistently low but non-zero number of participants who would choose *hate*.

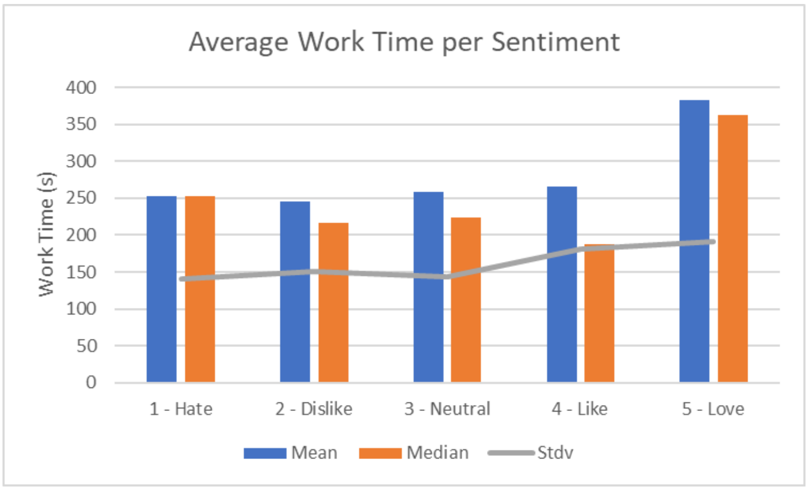


FIGURE 31 – AVERAGE PARTICIPANT WORK TIME PER SENTIMENT FOR ALL SYSTEMS

Another potentially interesting statistic to investigate is how long each participant spent on a task depending on what they rated it, shown in Figure 31. It is noticeable that significantly more time was typically spent on tunes that participants *loved*; Since some tunes were a significant amount longer than others, it is possible that listening to a longer tune increased a participant's familiarity with its musical ideas and made them more likely to enjoy it.

An alternative is that longer pieces of music simply have more potential for structure and development, which are factors that listeners might unconsciously seek in music.

Reject H_0 for $p < \alpha = 0.05$

H_0: Multitask gives identical results to RNN.
H_1: Multitask gives different results to RNN.
$Mann - Whitney\ Statistic = 177.000$
$p = 0.2641608486$
(do not reject H_0)

FIGURE 32 – MANN-WHITNEY TEST (MULTITASK AND RNN)

Reject H_0 for $p < \alpha = 0.05$

H_0: Multitask gives identical results to Human.
H_1: Multitask gives different results to Human.
$Mann - Whitney\ Statistic = 195.500$
$p = 0.4546190032$
(do not reject H_0)

FIGURE 33 – MANN-WHITNEY TEST (MULTITASK AND HUMAN)

Reject H_0 for $p < \alpha = 0.05$

H_0: RNN gives identical results to Human.
H_1: RNN gives different results to Human.
$Mann - Whitney\ Statistic = 173.500$
$p = 0.2327453694$
(do not reject H_0)

FIGURE 34 – MANN-WHITNEY TEST (RNN AND HUMAN)

To test the validity of the human evaluation results, statistical significance was calculated using the Mann-Whitney U test since the data was ordinal – rated on a Likert scale – and gathered using a between-subjects design. This gives a confidence value p for determining if two systems follow identical distributions.

Figure 32, Figure 33 and Figure 34 demonstrate that there is no statistical significance at the given α level, meaning we cannot confidently reject any of the given null hypotheses, therefore there is insufficient evidence to support the existence of a difference between the systems.

The conclusion to be taken from these results is that since the null hypotheses are accepted, the three systems can be considered as likely being from the same distribution, or in other words, that the baseline RNN and Multitask RNN give similar results to Human when their generated music is evaluated by a human.

Additionally, since the p value is considerably smaller when comparing RNN and Human than when comparing Multitask and Human – we are less confident that RNN and Human are similar than we are that Multitask and Human are similar – it can be inferred that Multitask does a better job of replicating the results of Human music.

Full tables containing results from human evaluation can be found in Appendix C.

8.3 Musical Analysis of Generated Output

A useful exercise to demonstrate the musicality of output from the models would be to annotate the score of a generated piece of music with section markers, chords and compose a bass line. Doing this would demonstrate how successfully a tune has captured the harmony of Celtic folk music (the bass line and chords should be intuitive to write) and how well the rhythm and structure make sense (the rhythm of the bass line and placement of section markers will also be intuitive to add). This was conducted with one piece for each RNN system, selected from those used in human evaluation.

A musical analysis for each piece was also performed, discussing concepts from Western music theory, particularly those typically found in Celtic music, in an attempt to understand composition choices which were made both during the generation and the annotation stages.

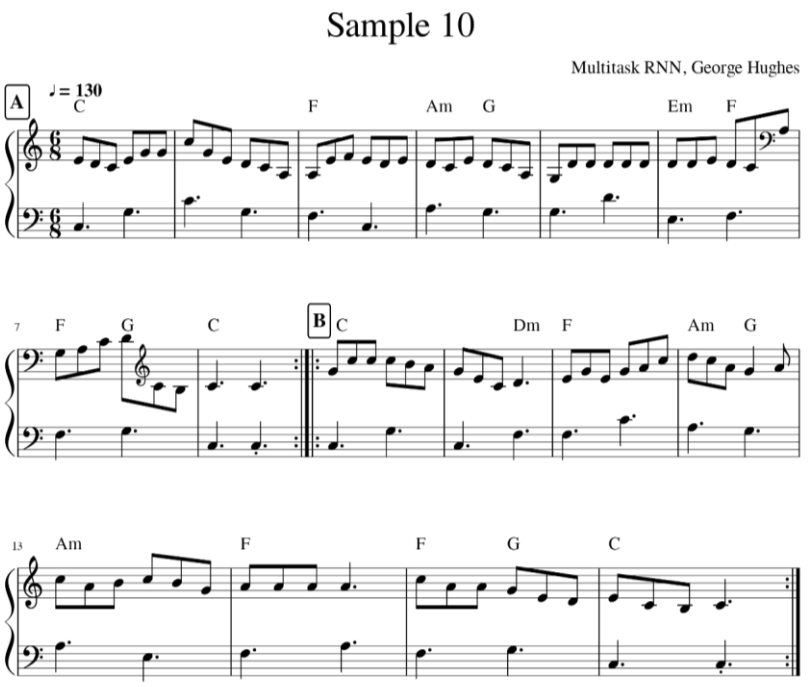


FIGURE 35 - SCORE OF A TUNE GENERATED BY MULTITASK RNN, MARKED UP WITH SECTIONS AND CHORDS WITH ADDED BASS PART

In Figure 35 is a tune generated by one of the Multitask RNN models. This piece was extremely fascinating as it both demonstrated a melodic theme and had very clear A and B sections. The theme was a simple case of heavily repeated notes (bars 5 and 14) but gave the piece a distinct character. Perfect cadences (chord V to I) can be found at the end of both sections and while the chords at the beginning and end of both sections match, the melody for the B section would not fit the same chords at two points: notably the D at the end of bar 10 (which can only comfortably sit with a C chord as a passing note in the middle of a phrase) as well as the entirety bars 13 and 14; This kind of evolution of the chords is normal for a B section in the style.

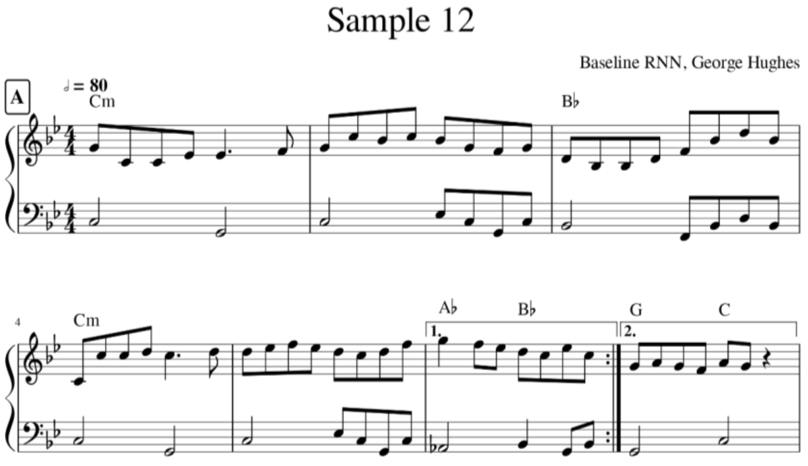


FIGURE 36 – SCORE OF A TUNE GENERATED BY BASELINE RNN, MARKED UP WITH CHORDS WITH ADDED BASS PART

Seen in Figure 36 is the score of a tune generated by a baseline RNN model. The piece sounds somewhat odd because the bars are grouped in threes instead of four; this is not unheard of in the style but it might be considered somewhat of a strong statement and is more difficult to make sound good – this is why it's unlikely the network learned to do this and rather happened to produce it by chance due to lack of rhythmic grasp. The grouping is reflected both in the chords and the lines (three bars to a line).

The piece is acceptable up until bar 7, which is played on the second repeat. The melody here implies a shift from C minor to C major, which is surprising and unnatural sounding, particularly when there exists only one bar of this change before the piece ends. For most of the piece before this, the harmony is fairly simple; chords i and VII are mostly arpeggiated by the melody and at bar 6, a “backdoor resolution” stepping through chords VI-VII-i was fitting; since these chords are characteristic of the Celtic style, it is fair to say that good harmonic choices were in effect for most of this piece.

Typically, pieces generated by the baseline RNN consisted of one section which would be repeated, then the piece would end. In the cases where this did not happen, the piece appeared to veer randomly without following a clear harmonic direction or forming distinct sections.

Rhythm was extremely difficult for the networks to grasp; even the models with the lowest perplexities from both systems produced music which didn’t have the right number of notes to a bar – when considering the information that the network had to work with however, it may not be reasonable to expect such a feature to be learnable or even computable simply from a sequence of prior symbols given the vanishing gradient. A common occurrence shared between both systems was the appearance of out of place sounding triplets.

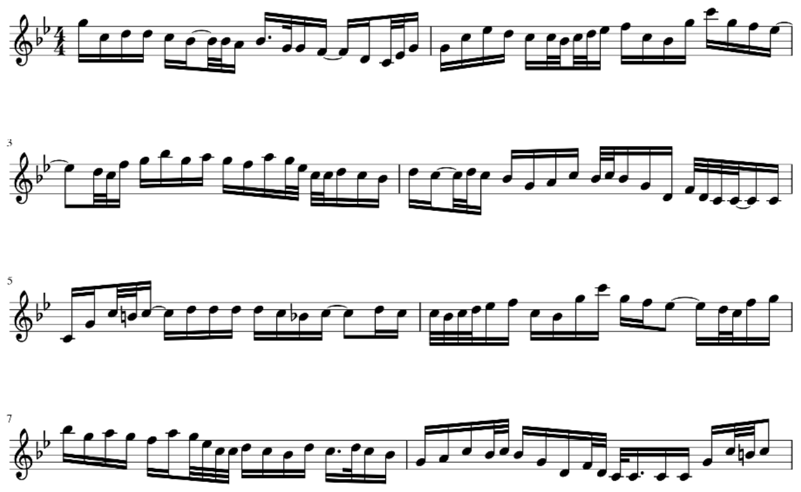


FIGURE 37 – CHAOTIC "TRANSCRIPTION" PIECE GENERATED BY BASELINE RNN

Occasionally, a model would produce a piece that seemed extremely chaotic, using extremely short time values for notes, but sounded like a transcription of how an experienced musician would play a piece, littered with ornamentation; it seems that a small number of "transcriptions" existed within the dataset leading to this rare behaviour. An example of a generated piece like this is scored in Figure 37.

9 CONCLUSION

9.1 SUCCESS

A clear success for this project was achieved in analysing performance using perplexity, particularly when comparing the overall performance of baseline RNN models to the performance of Multitask RNN models, the latter of which far better results could be achieved after a comprehensive tuning of hyperparameters. This went to show that the multitask model was statistically better at predicting music from the dataset than the baseline, and hence could be considered able to produce music that is more similar to that composed by a human.

As demonstrated in the musical analysis (section 8.3), the presence of A and B sections are found in music generated by Multitask models, while baseline RNN models are limited in this regard. Having sections is good composition practise in the Celtic style so it can be argued that since the Multitask model achieves using this feature, it is better at modelling the style.

As mentioned in section 6.1, it was expected that using structure as an auxiliary task would remedy the issues of "aimless" harmony found in related projects; after performing the musical analysis, the cadences that were found in the Multitask music compared against the baseline RNN made more musical sense, supporting this expectation.

The human evaluation results provided statistical evidence that Human music is closer to music generated by a Multitask RNN model than by the baseline RNN. The null hypotheses were accepted for all the defined cases and therefore the systems can be considered similar. While this is not the most useful result, it is still fairly optimistic and does indeed leave open the questions posed by the project.

9.2 LIMITATIONS

Regrettably, there were a number of major limitations in the work performed here. Namely, the failure the implement hierarchical neural networks compromised the original hypotheses; however, the initial requirements specified this architecture as a *could* using MSCoW, predicting that this outcome was a possibility.

Being unable to extract chord information created the limitation of only being able to test a Multitask network with a single auxiliary task. Being able to access chord information would very probably give better performance, since harmony is such an important facet of music. Accessing this information is difficult and potentially not possible to do effectively with a static computer algorithm, so human annotated scores with chords would be required to train a model to do this task; it is questionable to say that this type of information on any sufficiently large scale will be easily available in the near future.

The final main shortcoming here is the inability to confidently show statistical significance in the differences between RNN and Human or RNN and Multitask – doing so would have given confidence that the investment in extracting structure and building an auxiliary task results in being able to generate music with a similar likeability. Instead, there was not enough confidence to say that any of the systems give distinct results.

It may also be worth noting that beam search, which was one of the chosen generation methods in the plan, was dismissed in favour of random sampling and greedy search. It could have been interesting to see the results of such an algorithm; potentially the resulting music would have displayed different properties and may have acted as a medium between the two chosen methods.

9.3 Future Work

Future projects building upon this work would do well to look deeper into methods for studying human sentiments, particularly for music. This would ideally have the benefit of giving more definitive and conclusive results which would show preferable architectures for generating likeable music.

Leading on from this, various architectures could be explored for the purpose of music generation – such as the hierarchical neural network which theoretically shows promise but remained unimplemented here; it may even be possible to invent new architectures for this specific domain. The types of input data could also be experimented with if dataset availability permits, using waveforms, MIDI music or defining a new format with the goal of being easily learnable by a machine – possibly extending this to allow polyphonic music.

As mentioned in limitations, it would also be interesting to see development of a model with the specification of being able to tag music with chords, using either deep learning or by devising an algorithm which can extract such information in a similar manner to how a human would compose chords.

10 REFERENCES

Bowman, S. R., Angeli, G., Potts, C. & Manning, C. D., 2015. *A large annotated corpus for learning natural language inference.* Lisbon, Association for Computational Linguistics.

Cowan, M., 2018. *Music Composition Using Deep Learning,* Edinburgh: s.n.

Goldberg, Y., 2017. *Neural Network Methods for Natural Language Processing.* s.l.:Morgan & Claypool.

Granroth-Wilding, M. & Steedman, M., 2014. A Robust Parser-Interpreter for Jazz Chord Sequences. *Journal of New Music Research,* 43(4).

Haykin, S., 2009. *Neural Networks and Learning Machines.* 3rd ed. Ontario: Pearson.

Hochreiter, S., 1998. The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* Volume 6, pp. 107-116.

Hochreiter, S. & Schmidhuber, J., 1997. Long Short-term Memory. *Neural Computation,* Volume 9, pp. 1735-80.

Kocmi, T. & Bojar, O., 2017. An Exploration of Word Embedding Initialization in Deep-Learning Tasks. *CoRR,* Volume abs/1711.09160.

Mehr, S. A. et al., 2018. Form and Function in Human Song. *Current Biology,* 28(3), pp. 356-368.

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. Efficient Estimation of Word Representations in Vector Space. *arXiv e-prints.*

Pennington, J., Socher, R. & Manning, C. D., 2014. *GloVe: Global Vectors for Word Representation.* Doha, Association for Computational Linguistics.

Ruder, S., 2017. *An Overview of Multi-Task Learning in Deep Neural Networks,* s.l.: eprint arXiv:1706.05098.

Sears, D. R. W., Korzeniowski, F. & Widmer, G., 2018. *Evaluating language models of tonal harmony*. Paris, arXiv:1806.08724.

Serban, I. V. et al., 2015. *Building End-To-End Dialogue Systems Using Generative Hierarchical Neural Network Models,* s.l.: arXiv:1507.04808.

Serrano, L., 2016. *YouTube.* [Online] Available at: https://www.youtube.com/watch?v=BR9h47Jtqyw [Accessed 17 October 2019].

Sordoni, A. et al., 2015. *A Hierarchical Recurrent Encoder-Decoder For Generative Context-Aware Query Suggestion.* s.l., eprint arXiv:1507.02221.

Sturm, B. L., Santos, J. F., Ben-Tal, O. & Korshunova, I., 2016. *Music transcription modelling and composition using deep learning.* s.l., arXiv:1604.08723.

Sun, M. et al., 2016. *Chinese Computational Linguistics and Natural Language Processing Based on Naturally Annotated Big Data.* 1st ed. Yantai: Springer International Publishing.

Trevett, B., Reay, D. & Taylor, N. K., 2018. *The Effectiveness of Pre-Trained Code Embeddings.* s.l., s.n.

Vaughn, K., 2000. Music and Mathematics: Modest Support for the Oft-Claimed Relationship. *The Journal of Aesthetic Education,* 34(3/4), pp. 149-166.

Yaov, G. & Søgaard, A., 2016. *Deep multi-task learning with low level tasks supervised at lower layers.* Berlin, Association for Computational Linguistics.

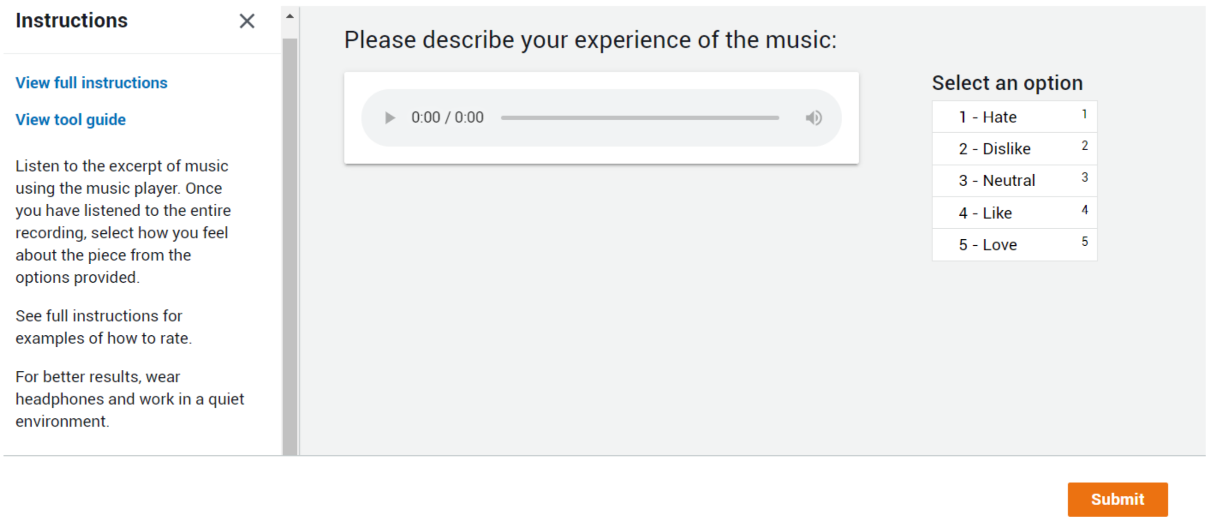
11 APPENDICES

11.1 APPENDIX A

Model Name	Hidden Size	Embeddings Size	Embeddings Dropout	LSTM Dropout	Number Hidden Layers	Aux-task Weight	Perplexity	Aux-task Perplexity
multitask_08	256	256	0	0	1	0.005	4.253	2.99
multitask_07	256	256	0	0	1	0.05	4.312	2.074
multitask_05	256	256	0.1	0.2	1	0.05	4.352	2.126
multitask_10	256	256	0	0	1	0.0005	4.352	6.756
multitask_04	256	256	0.1	0.2	1	0.1	4.359	1.582
multitask_03	256	256	0.1	0.2	1	0.25	4.382	1.336
multitask_02	256	256	0.1	0.2	1	0.5	4.402	1.203
baseline_08	256	256	0.1	0.2	2		4.417	
multitask_06	256	256	0.1	0.2	1	0	4.435	9.738
multitask_11	256	256	0.1	0.2	1	0.0005	4.506	6.77
multitask_00	256	256	0.1	0.2	1	1	4.574	1.183
multitask_09	256	256	0.1	0.2	1	0.005	4.618	3.066
baseline_07	256	256	0.1	0.2	1		4.814	
baseline_09	256	256	0	0	1		4.853	
baseline_10	512	512	0.3	0.5	1		4.873	
baseline_06	128	128	0.1	0.2	2		5.203	
baseline_05	128	128	0.1	0.2	1		5.205	
baseline_02	128	128	0.3	0.5	1		5.709	
baseline_03	64	64	0.3	0.5	2		6.502	
baseline_04	128	128	0.6	0.7	1		6.604	
baseline_00	64	64	0.3	0.5	1		6.606	
baseline_01	32	32	0.3	0.5	1		8.217	
multitask_01	256	256	0.1	0.2	1	5	8.434	1.308

FIGURE 38 – TABLE CONTAINING ALL TRAINED MODELS, THEIR HYPERPARAMETERS AND PERPLEXITIES, SORTED BY PERPLEXITIES, ASCENDING

11.2 Appendix B


Instructions

View full instructions

View tool guide

Listen to the excerpt of music using the music player. Once you have listened to the entire recording, select how you feel about the piece from the options provided.

See full instructions for examples of how to rate.

For better results, wear headphones and work in a quiet environment.

Please describe your experience of the music:

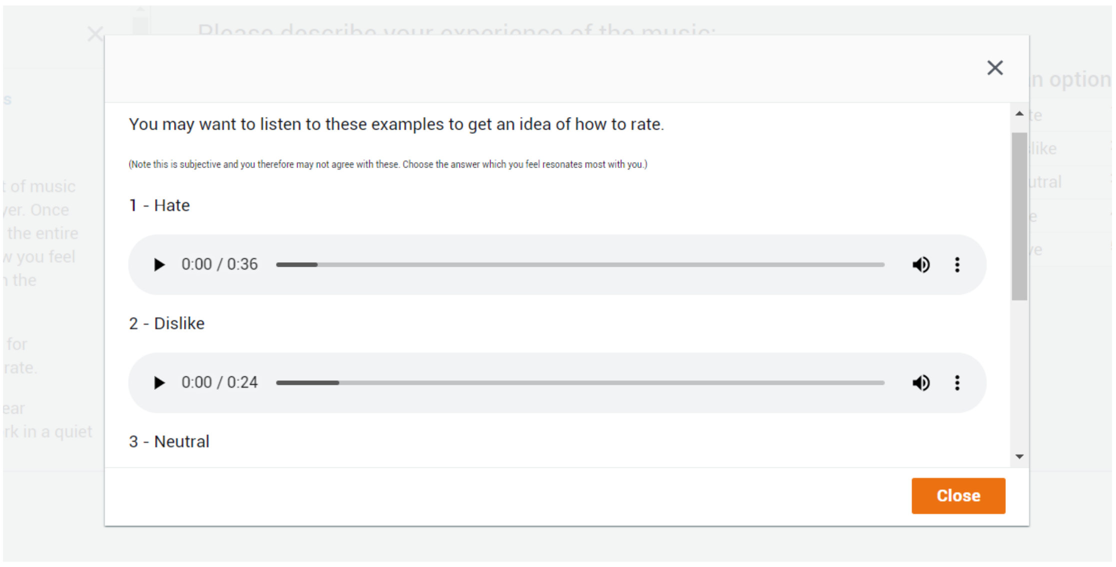
0:00 / 0:00

Select an option

1 - Hate	1
2 - Dislike	2
3 - Neutral	3
4 - Like	4
5 - Love	5

Submit

FIGURE 39 – SCREEN CAPTURE OF AMAZON TURK INTERFACE WHICH WAS PRESENTED TO PARTICIPANTS


You may want to listen to these examples to get an idea of how to rate.

(Note this is subjective and you therefore may not agree with these. Choose the answer which you feel resonates most with you.)

1 - Hate

0:00 / 0:36

2 - Dislike

0:00 / 0:24

3 - Neutral

Close

FIGURE 40 – SCREEN CAPTURE OF AMAZON TURK INTERFACE WHEN PARTICIPANT VIEWS FULL INSTRUCTIONS

11.3 APPENDIX C

	Baseline RNN	Multitask RNN	Human
1 - Hate	1	0	1
2 - Dislike	3	3	2
3 - Neutral	6	4	3
4 - Like	5	8	9
5 - Love	5	5	5

FIGURE 41 – TABLE CONTAINING COUNTS FOR EACH RESPONSE FOR EACH SYSTEM

11.3.1 BASELINE RNN

Input.audio_url	Answer.listener-experience.label
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune1.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune1.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune1.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune1.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune2.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune2.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune2.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune2.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune3.mp3	1 - Hate
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune3.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune3.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune3.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune4.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune4.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune4.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune4.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune5.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune5.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune5.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system1/tune5.mp3	5 - Love

FIGURE 42 – TABLE CONTAINING ALL RESPONSES FOR SYSTEM 1

11.3.2 MULTITASK RNN

Input.audio_url	Answer.listener-experience.label
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune1.mp3	2 - Dislike

https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune3.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune5.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune2.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune2.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune4.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune4.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune1.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune2.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune3.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune3.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune4.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune5.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune5.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune5.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune1.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune1.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune2.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune3.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system2/tune4.mp3	5 - Love

FIGURE 43 – TABLE CONTAINING ALL RESPONSES FOR SYSTEM 2

11.3.3 HUMAN

Input.audio_url	Answer.listener-experience.label
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune5.mp3	1 - Hate
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune3.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune5.mp3	2 - Dislike
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune2.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune3.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune5.mp3	3 - Neutral
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune1.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune1.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune1.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune2.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune2.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune3.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune3.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune4.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune4.mp3	4 - Like
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune1.mp3	5 - Love

https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune2.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune4.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune4.mp3	5 - Love
https://www2.macs.hw.ac.uk/~gmh1/honours/music_samples/system3/tune5.mp3	5 - Love

FIGURE 44 – TABLE CONTAINING ALL RESPONSES FOR SYSTEM 3

11.4 Appendix D

Link to the GitHub repository for this project: https://github.com/georgehughes1998/Honours