University of Glasgow | School of Computing Science

Honours Individual Project Dissertation

Learning Domain-Specific Structured Embeddings for Symbolic Music with LSTMs

Salman Mohammadi
March 27, 2019

Abstract

Language, among many others, is a strong indicator for intelligence, and complex languages underpin a huge number of vital high-level communication processes in what we consider intelligent life. Music is one such language, with many nuanced and intricate elements that may involve cultural and social context, musical ability, personality, sex, and memories, which influence the way a piece of music is created and understood by listeners. Deep learning methods which would allow for automatic interpretation and generation of music might allow us to change the way we interact with each other and the world around us in entirely novel ways, as well as the possibility of applying these techniques to other languages, even entirely new ones. One example could involve tailoring automatically generated music for application in music-assisted therapy. This project will investigate deep learning approaches for automatically generating music in a symbolic representation (such as MIDI), and also learning domain-specific embeddings for symbolically represented music. The aim of the project is to conditionally create music which embodies the compositional style for a particular composer, and we aim to evaluate this with user studies and qualitative metrics.

Education Use Consent

Acknowledgements

I would like to thank my supervisor, Bjorn Sand Jensen, for his excellent mentorship and the continual advice and support he provided through this project.

Contents

1 Introduction

In this dissertation we present a novel approach for conditionally generating musical sequences. We show how this method improves upon a baseline for conditional sequence generation, and evaluate the method in a number of ways to demonstrate its ability to model salient features in symbolic music sequences.

1.1 Motivation

Music has its beginnings in great antiquity (Montagu 2017), and there has been significant study in grounding the psychology of musical perception in an evolutionary basis (Mithen 2006; Huron 2012; Darwin 1871). In one review, McDermott and Hauser (2005) discusses the prevalence of neural mechanisms for certain aspects of musical structure. This might explain why aspects of music are universal across cultures, such as detecting tonal hierarchies, and an enjoyment of rhythm and dancing. Despite its origins and precise functions in human psychology remaining elusive, it's clear that music is innate experience in all humans. This is only cemented by the prevalence of systems in the human brain specialised towards processing music.

Language is an example of one of the many ways in which the human brain is highly specialised (Brown and Hagoort 2000). There are many similarities between music and language (Patel 2003), with music relying on specific structural and syntactical elements which seem to be omnipresent across different cultures and traditions (Drake and Bertrand 2001). Such elements utilise repetition and expectation as fundamental mechanisms which allow humans to learn this grammar of music implicitly through listening to it. These mechanisms are essential for conveying a depth of information and creating emotion and meaning in its listeners and performers (Vuust and Frith 2008; Huron 2006). Understanding music in this manner is still as unavoidably vague, but we can now consider that music is able to represent ideas and emotions through certain kinds of salient features; language-like sequences which employ elementary musical principles to structure sounds and create incredible depth of meaning and rouse its listeners.

With this consideration, we can argue that some of the high-level, abstract concepts which we attribute to certain pieces of music can be decomposed into these features. A musical piece which is described as "uplifting" could be thought of as all of the temporal dependencies that span a piece. It would demand a relatively skilled human musician if we were interested in composing musical pieces which specifically embody certain high-level abstract concepts, and it's likely that the human would draw on a deep base of individual and cultural musical knowledge and experience.

Across many fields which benefit from the application of artificial intelligence, all of them have the same underlying goals; to create agents which can match or surpass human level intelligence or ability in some aspect. With advances in deep learning making possible neural architectures of considerable depth, we are seeing networks of sufficient sophistication in design to encode latent representations of complex and high-dimensional temporal data Roberts et al. (2018).

A model which could capture the latent features that make up some of the higher-order, abstract terms that describe music would contribute greatly to a number of fields. One of the largest issues in neural language models lies in the varying scales of temporal dependencies which ubiquitous in almost every human language, and particularly one such as music. By developing new methods for conditional music generation, we can see that the unique problems present in modelling such a nuanced language demand solutions which will greatly contribute to the wider field of natural language processing.

Furthermore, the breadth of scope for application of such a model is vast: a system which could learn to generate musical sequences which elicits some emotional response from its listener could revolutionise music-assisted therapy, an application that allows humans to play alongside a trained network in a style from their favourite composer, or even a completely novel system which can generate music by being conditioned on images, videos, or forms of written poetry.

1.2 Aims

In this dissertation we focus on the application of modelling high-level concepts for generating sequences of music. A necessary first step for this is to be able to generate realistic music, which is able to capture some significant musical structure. We then aim to build upon this model to condition generation of music on a specific label, from a set of abstract labels which are used to describe music. Explicitly, the goals of this dissertation are outlined:

- Using deep learning to create realistic music.
- Using deep learning for conditionally generating music, based on some abstract term that can describe a sequence of music.

To demonstrate these goals we develop a novel deep learning method, and evaluate our method directly with respect to each of the aims of the project. Our evaluations will involve vital qualitative studies which use human participants, and also quantitative experiments to test our hypothesis directly.

1.3 Summary

The motivations and aims for this project were stated in this chapter, and the following chapters of this dissertation are detailed:

- Chapter 2 discusses relevant prior work, and covers background knowledge.
- Chapter 3 reviews datasets and data formats.
- Chapter 4 provides an in-depth analysis and refinement of our research hypothesis.
- Chapter 5 details the methodology used to design our model.
- Chapter 6 gives an overview of the implementation process.
- Chapter 7 analyses the results from the evaluation.
- Chapter 8 concludes the project and provides possible extensions.

2 | Background

Automatic music composition is a broad field which is as diverse in its range of methods as it is applications. For relevance to this dissertation, this chapter will outline existing research in approaches which use neural networks to implicitly learn patterns from data. In the following sections we cover prior work which forms the backbone for our approach, and aim to address these approaches with respect to the aims of this project.

We assume the reader has some familiarity with neural networks, but refer Goodfellow et al. (2016) and Nielsen (2015) as excellent resources if not.

2.1 Generating Realistic Music

This section covers relevant research in addressing whether neural networks are able to create realistic music, and also introduces background topics which are key to understanding this project.

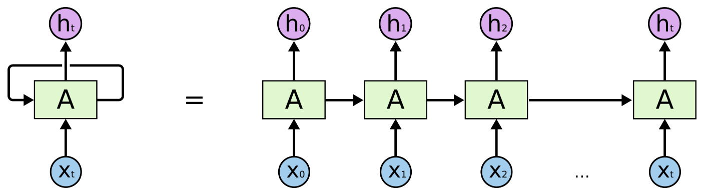
2.1.1 Background

Recurrent neural networks (RNNs) (Karayev et al. 2014) are able to capture information about sequences. They do this by persisting data from previous inputs, as *internal states*. Figure 2.1a demonstrates how recurrent networks operate on sequences. Given an input sequence x of length t, $x_0, x_1, ..., x_t$, a recurrent network produces a number of *hidden states*, $h_0, h_1, ..., h_t$. These hidden states are powerful and compact representations of their inputs.

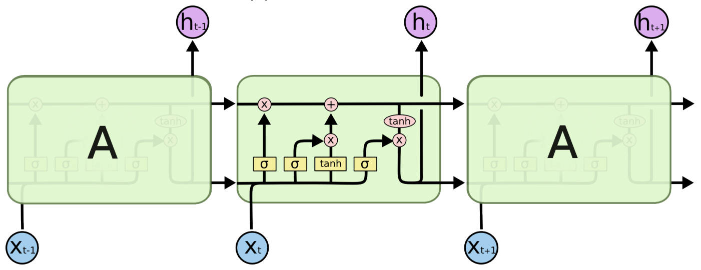
In principal, recurrent neural networks fail to capture long-term dependencies due to the issue of *vanishing gradients* (Hochreiter et al. 2001), where error flow between two temporally distant events during training vanishes quickly. *Long-Short-Term-Memory networks* (LSTMs) (Gers et al. 2000) are specialised recurrent networks which are well adapted to modelling information over arbitrary time intervals. These networks are composed of LSTM cells, which are shown in Figure 2.1b. These cells learn to store information over arbitrary time intervals by regulating flow in and out of the cells.

2.1.2 Prior Work

Some of the earliest approaches for generating music with neural networks by Lewis (1988), involved using gradients in multi-layer perceptron networks to compose novel musical motifs. Around the same time, Todd (1988) explored the use of recurrent neural networks for generating music sequentially, or note-by-note. The method utilised recurrent connections to store information about sequences of music, and was able to produce new sequences which had sensible local contours, but the technique suffered greatly from a lack of global cohesion as outlined by Mozer (1994).



(a) A recurrent network



(b) An LSTM cell

Figure 2.1: *Recurrent neural network diagrams. (a) shows an unrolled "vanilla" recurrent neural network which produces a number of hidden states for a given input. (b) shows a chain of LSTM cells which act similarly, but are also able to learn the most relevant information to store. (Olah 2015)*

Music is inherently reliant on repetition and requires structure at multiple levels, and a first approach by Eck et al. used LSTM cells to better capture long term dependencies. Liang et al. (2017) used a deep LSTM model which was able compose music in the style of Bach. This sequential model predicted music scores using a novel symbolic encoding for music, and vitally demonstrated that neuron activations in such a trained model correspond to musical theory concepts. However, this approach benefits from solving a constrained problem. Liang et al. (2017) only attempted to model the style from one composer, and also used a very homogeneous subset of that composer's work. Furthermore, Bach is described to have an algorithmic compositional style (Sealy), which no doubt lends a hand in the performance of the model.

The work done by Oore et al. (2018) demonstrated an approach which simultaneously captured the musical structure present in the classical piano dataset it was trained on, and the performing styles of the human composers which had played the pieces of the dataset. The model was able to exhibit a pleasing amalgamation of the various performing and composer styles it had been trained on, but, again, the methods used would be unable to isolate specific performing or compositional styles.

Contrary to the methods listed above, a recent work by Brown and Hagoort (2000) used a *Transformer* (Vaswani et al.) network which demonstrated state-of-the-art performance in sequence modelling. Samples produced from this network exhibited compelling structure of timespans of up to a minute, and the model was able to coherently extrapolate on a given musical motif, indicating an ability of the model to represent long-term dependencies in music at several levels of structure. Quite broadly, a transformer network operates using attention mechanisms (Bahdanau et al. 2014), which allow a model to learn the most relevant and important features of a given input sequence, similar to the attention mechanisms in human vision systems. While this network is able to capture dependencies across timespans several times larger than in previous efforts, it's unclear how such a model could be adapted to capture specific high-level attributes of music.

2.1.3 Convolutional Networks

Convolutional neural networks (Rawat and Wang 2017). are biologically inspired models which utilise the basic convolutional operator, and is primarily used in images. Convolutions aim to extract features from their inputs by moving a *sliding window* across their inputs. The window moves across a certain *stride length* and applies a *kernel* to the set of values formed by each window, and a single convolution is shown in 2.2.

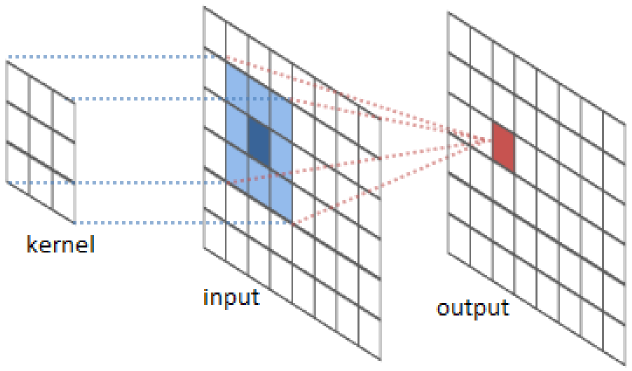


Figure 2.2: Olah (2014)

Convolutional networks model music directly as raw waveforms, i.e. as digital encodings of audio, which is in contrast to the symbolic representations used in sections up to now. Modelling music as raw audio waveforms is a far more demanding task, since music in the raw audio domain must be sampled many tens of thousand times a second, and models which operate at this level predict one waveform at a time. To capture any kind of musically relevant structure, any such model would require a very high representational capacity. Convolutional networks are severely limited in this application due to their small *receptive fields* (Donahue et al. 2018), which dictate the amount of information any particular convolutional filter has access to.

An updated approach by Donahue et al. (2018) introduced *dilated convolutions*, as shown in Figure 2.3, which allow for larger receptive fields to better contextualise information. With the advent of deep autoregessive architectures (Gregor et al. 2013), this presented opportunities to tackle the complexities of modelling raw waveforms. Donahue et al. (2018) and van den Oord et al. (2016) demonstrate models which are able to predict raw audio, one waveform at a time. Their experiments show that when applied to music, the models are able to capture local structure, but

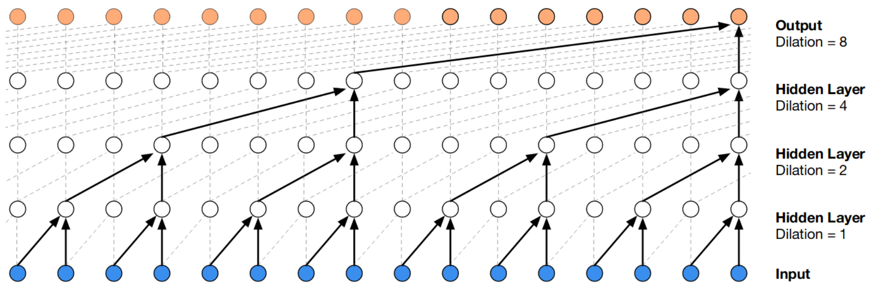


Figure 2.3: Dilated Convolutions van den Oord et al. (2016)

performed worse than the recurrent models discussed earlier, and were unable to display any kind of interesting coherence beyond a timescale of seconds.

Overall, the performance of convolutional models which operate directly on audio signals suffers due to the scale of the problem, and there are many necessary steps involved in modelling for the raw audio domain which need to be solved before being able to capture temporal structure at such a level.

2.2 Conditionally Generating Music

Little work exists specifically for conditionally generating music. This section will give an overview of any existing approaches that are similar, but also methods which show good potential for being adapted to the use.

Roberts et al. (2018) detail *Music-VAE*, a Recurrent Variational Auto-encoder (Bowman et al. 2015) model which is able to model long-term dependencies in symbolic music sequences. It uses an LSTM network to encode MIDI sequences into a latent distribution (cite variational auto-encoder), which is then sampled from to create a *latent vector*. Their main contribution lies in using a a hierarchical decoder which employs an intermediate recurrent network to decode this latent vector into inputs for the decoder LSTM network. This intermediary network imposes long-term structure in the musical sequences. While this model is somewhat removed from the task of conditional music generation, its architecture is well-suited for adaptation to the task.

Variational auto-encoders are powerful models which are capable of learning highly efficient data representations. For example, it's possible to sample latent vectors for all musical sequences which exhibit a certain style or property, e.g. all sequences which belong to a composer, or which are labelled by humans with a specific emotion. By averaging all of these latent vectors, we can produce an *attribute vector* (Donahue et al. 2018) as representative of that particular class. There are numerous inventive ways in which these attribute vectors can be used; given the latent vector for a sequence which is composed by X, we can subtract the attribute vector for X, and add the attribute vector for Y. Decoding this result might result in a realistic attribute transfer of the sequence.

A Universal Music Translation Network (Mor et al.) is an approach using a Wavenet Auto-encoder Engel et al. (2017) to conditionally transfer between musical domains. Here, the author demonstrates an encoder network which is able to learn representations of its input that are universal for their domains. They achieve domain transfer by training several different decoder networks, which each learn to decode this universal encoding into separate domain-specific outputs. The ability of this network to learn abstract, semantic representations of raw audio is particularly motivating for this project. A network architecture which could learn higher-order embeddings for musical sequences, and structure these embeddings in a manner which is relevant for modelling certain abstract terms, would potentially allow for conditionally generating coherent music that embodies specific attributes.

2.3 Transfer Learning

Transfer learning (Caruana 1997) is a machine learning method which involves using knowledge learned from one task to improve performance on a second, related task. Venugopalan et al. (2016) investigated the use of pre-training LSTMs on text corpora for improveing performance in generating natural language descriptions of videos, and Gulcehre et al. (2015) showed significant improvement in language translation tasks by taking advantage of widely available mono-lingual corpora. Neural models in such tasks are usually limited by the lack of availability of parallel, multi-lingual datasets, and the approach is promising for application to music. A significant barrier to developing conditional music generation models is the lack of good-quality, homogeneous, and labelled datasets. A model which could learn relevant information from unlabelled datasets could be applied to a conditional generation task, and demonstrate improved performance by fine-tuning learned musical knowledge.

2.4 Summary

In summary, we have introduced important concepts for the remainder of this dissertation, and, while investigating relevant prior work, we have identified a significant lack of research into conditional music generation. Despite this, we have outlined several key contributing papers that will form the backbone of this research, and also recent work in variational inference and transfer learning, which present promising opportunities for novel applications in conditional music generation.

3 Data

Deep learning approaches are only relevant in the context of the data they learn from. Even the most powerful algorithms require a vast amount of data, and a problem's complexity can be magnified by the scale of the dataset required. In this chapter we discuss an appropriate format for our data, and review the dataset we will be using with respect to the goals of the project.

3.1 Digital Music Formats

Music exists primarily in audio, and is also represented in many different formats at varying levels of abstraction. We can consider these formats with respect to our first goal of generating realistic music. Any relevant evaluations of a model which can generate realistic music should involve listening to the music itself. However, all representational formats of music require certain steps to transform a particular representation into sound, and different representations also omit features of music which must be re-added to create sounds. With this consideration, the following sections will review the availability, prior use, and relevance of datasets in different formats for realistic music generation.

3.1.1 Audio files

Audio recordings are one of the most prevalent forms of music format, and account for more than half of all global music revenue (IFPI 2018). Digital audio is encoded as a sound wave, comprised of a continuous sequence of samples which are taken in fixed intervals, e.g. CD audio format takes samples 44,100 times every second. While raw audio signals are the most accurate digital music format, modelling musical audio signals is far more difficult than symbolic representations of music. A network which could generate music in the audio domain would need a very large representational capacity, and such a complex network would require a large amount training data. Most publicly available audio datasets are distributed in WAV or MP3 format which use up to 10MB of disk space per minute of audio. We can see that attempting to at least generate realistic music in the audio domain is a task which would demand a large amount of computational resources. Coupled with the fact that modelling raw audio presents an additional layer of complexity, we proceed by considering alternative formats.

3.1.2 Notated music

Musical scores, or sheet, are a form of musical notation that use symbols to denote notes, timing, meter, and stylistic information. Sheet music can be thought of as instructions on how to play a piece of music, rather than a specific performance of one. This distinction is important, as sheet music is often not followed exactly. Figure 3.1 shows an example of many techniques used by musicians which aren't precisely described in sheet, such as rubato, swing, and dynamic markings. ABC notation Walshaw is a text-based musical notation system, which captures similar information about musical scores by using the letters A to G to represent notes. ABC music

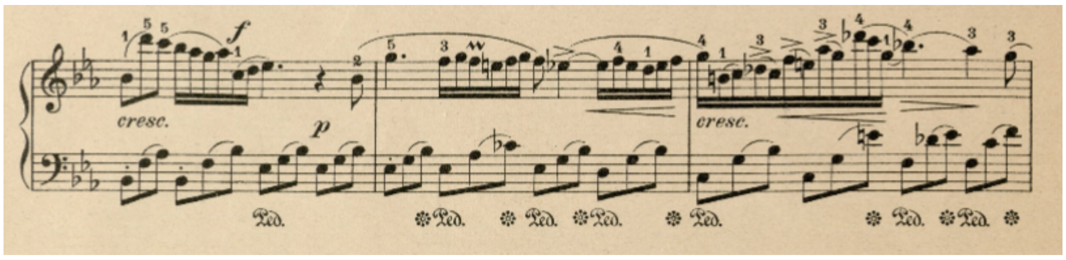


Figure 3.1: *Excerpt from the score of Chopin's Nocturnes Op. 9 No. 2. The sheet markings show examples of inexact musical notation such as trill, forte, piano, and diminuendo and crescendo.*

files are around six orders of magnitude smaller than raw audio files since they only consist of ASCII characters, but are unable to be listened to directly, and must be converted to an intermediary format before being converted to audio. We can see that this format is a highly abstract representation of music, and relies on the musical knowledge of a trained human to effectively convert aspects of music into sound. Human-performed musical pieces are often full of subtlety and nuance, which musical score is unable to represent precisely. These nuances are essential for conveying expressive musical ideas and creating realistic music. A suitable format for this project might lie somewhere between musical score and raw audio.

3.1.3 MIDI files

The Musical Instrument Digital Interface (MIDI) (Swift 1997) format records information from specialised MIDI digital instruments. The MIDI format only captures information about the notes, or pitches, and how loud each note or pitch was, and also the controls of a specific instrument, e.g. pedalling for a piano. While MIDI files are able to capture precise timing and velocity [1], the format doesn't encode specific aspects of musical performances such as timbre. However, it benefits from being a highly compressed representation, and MIDI files are usually around three orders of magnitude smaller than raw audio files. We can consider the MIDI format to be a sequence of events which can be re-played using audio samples.

The missing features of musical scores which are often vital to the enjoyment of a musical piece, such as expressiveness and dynamics, are able to be concisely represented with the MIDI format. Furthermore, the relative ease with which MIDI files are able to be distributed digitally means that there is a prevalence of MIDI datasets.

Given the appropriate choice of MIDI file format, we review in the following section the dataset which we plan to use.

3.2 Dataset

We require an appropriate dataset to allow us to generate realistic music, but also condition that generation on some descriptive, abstract label. However, there is no such prevalence of large, labelled MIDI datasets. The Lakh MIDI Dataset (Raffel 2016) consists of provides a subset of 45,129 MIDI files which have been matched to entries in a dataset detailed by Schindler et al. (2012). While it's possible to retrieve genre labels for these files, the dataset still suffers from lack of homogeneity due to being collected from numerous publicly available sources, which also makes the likelihood of most of the data being high quality quite low.

[1]The term velocity originates from how fast a piano key is struck.

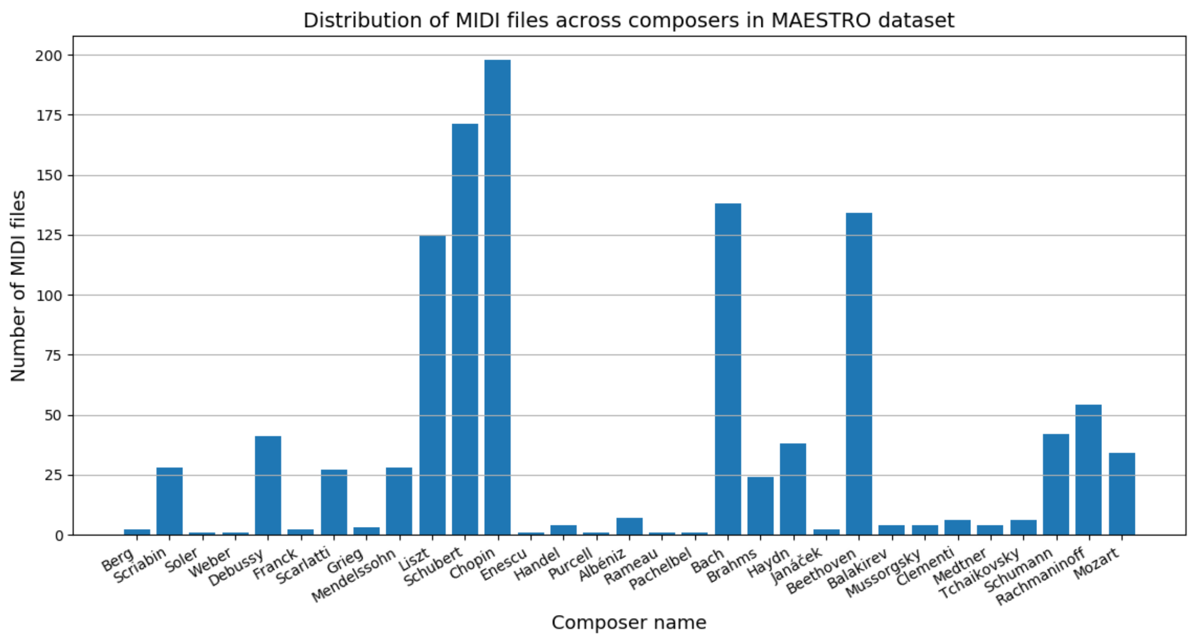


Figure 3.2: *Displays distribution of files across composers in the MAESTRO dataset. There's an abundance of composers from several distinct classical music periods.*

The MAESTRO (Hawthorne et al. 2018) dataset provides roughly 1,200 MIDI files curated from digital piano competitions. These MIDI files have been recorded using specialised grand acoustic pianos, which are capable of so precisely capturing piano performances that they allow the entire judging process to be completed remotely. Just as importantly, each MIDI file is labelled with the composer of the piece being performed, which provides us with the necessary information to design an approach which might be able to capture musical information about specific composers.

It's vital to note the importance of a homogeneous and high quality dataset, as any model that we train is likely to capture the characteristics of the dataset it has been trained on. The MAESTRO dataset is especially suitable for our task for numerous reasons. Firstly, the dataset is all classical, solo piano music. This helps the model learn a coherent representation of musical sequences, and constrains the problem in a way which doesn't subtract from our goals. Even though multi-instrumental music is wholly ubiquitous now, a model which can capture the characteristics of solo classical piano music is still very sufficient for this project. Additionally, all MIDI files in the dataset are performances by expert human musicians. If our goal is to create realistic music, it helps that the music the model learns from is indeed real music which is composed by renowned artists, and performed by humans with exceptional musical skill.

3.2.1 Compositional Style from Data

We aim to use our model to learn multiple specific compositional styles, and be able to be conditioned on a single compositional style to generate distinct musical sequences. The MAESTRO dataset comprises of 31 composers, as seen in Figure 3.2. We selected four composers from set of 31 which were from distinct classical music periods, and had most number of pieces in the dataset. The composers were Bach, who composed in the Baroque era, Chopin, who composed in the Romantic era, Beethoven, who composed in the Classical era, and Debussy, who composed in the Impressionist era.

3.3 Summary

In this chapter we have reviewed data formats and data sources. We have contextualised their applicability to the aims of this project, and as a result, have narrowed our goals to the application of conditional symbolic music generation. We will use a subset of the MAESTRO dataset to condition our generation on composer style. Thus, given our choice of data format and dataset, we can refine our goals to better reflect work in relevant literature and availability of data.

4 Analysis

In this chapter we will update our original aims in light of relevant prior work, available data, and our analysis of data formats.

The dataset we use constrains our approach in several ways. We have analysed a high-quality and homogeneous classical piano dataset in the MIDI format, and designed a subset of this data which consists of four distinct composers. This has narrowed our problem domain and allowed us to better design our aims for the project. After reviewing current literature in deep learning for music generation, we have identified two research questions which will serve as the objectives for this dissertation:

4.1 Generating realistic music

Deep learning and neural networks have been shown to perform very well in capturing local contours of music in both symbolic form and raw waveform. Generating realistic music is a necessary pre-requisite goal before being able to condition our model on a specific composer's style, and while several sophisticated approaches exist (Dieleman et al. 2018; Huang et al. 2018), our overarching goal is to directly demonstrate our overall hypothesis of conditional music generation.

A first natural step, then, is to start with an approach that is simple to implement, but generates musical sequences with coherence on a level which can exhibit compositional style. Recurrent networks have seen great success in relevant literature for modelling symbolic MIDI music. A natural first step for this dissertation, is therefore to design a recurrent neural network architecture which is able to generate realistic music sequences that are cohesive on a timescale that has already been demonstrated. We will base our research on existing approaches BACKGROUND, and if successful, will provide us with a model which we can build upon for our second research goal, and also a baseline to compare our approach to.

4.2 Conditionally generating by composer style

The model developed in the first research objective would create music which is an amalgamation of the four different composers' styles it had been trained on. The model in its current state would be unable to capture the specific characteristics of a single distinct compositional style. Thus, a more sophisticated model which utilises specific information learned from the labelled dataset detailed in Section 3, could potentially generate music by being conditioned on a composer.

While this goal would benefit greatly by reusing work as as in (Roberts et al. 2018), we aim to start more simply in order to directly demonstrate our hypothesis. To do so, we plan to modify the model developed in the first hypothesis, and utilise existing work in classification of time-series, transfer learning, and multiple-output networks. At a high level, we aim to design an objective function which we can optimize to represent the different compositional styles as discreet clusters in our model's internal states. By doing so, we separate the different salient features which make up each compositional style, and encourage our model to learn meaningful representations of its input data.

More precisely, our goal is to impose constraints on the model's internal states which *disentangle* the different compositional styles, and encourage our model to learn a discreet embedding for each composer. If we can represent any sequence by a given composer as a point on our model's internal states, we can aim to optimize an objective function which:

- Minimise the distances between all points for a single composer.
- Maximise the distances between points which belong to different composers.

The developed model will be evaluated by using human experiments, where we will use discriminating testing (Lawless and Heymann 1999) to determine if the model's disentangled latent representations are able to be perceived in music it generated. We will also to analyse the internal states of the model, and develop metrics specifically for evaluating our model directly with respect to this research goal.

5 Methodology

This chapter will outline our overall approach for demonstrating each of our hypothesis. We cover our model architecture for generating realistic music, and then introduce a novel recurrent network for learning discrete representations of compositional style. Finally, we examine data representations for our models and design suitable encodings for our MIDI data.

5.1 Realistic Music

We re-use an existing approach for expressive piano music detailed in (Oore et al. 2018), and our model in this section is an implementation of the architecture discussed here. For our base model, we use a deep LSTM recurrent network highlighted in Figure 5.1, which we will refer to by component number for this section. Quite broadly, our LSTM operates on an input sequence by producing a final hidden state from LSTM Cell 3. This final state is fed into a feed-forward layer[1], which uses a *softmax* activation function Goodfellow et al. (2016), to predict a probability distribution over the model's encoding vector, which we discuss in detail in a later section.

This probability distribution is used to compute the *cross-entropy loss*, or *log-loss* of the model for a certain input sequence. The cross-entropy loss indicates how much the model's softmax output has diverged from the training label, and this value is used to optimize our model.

To generate music sequences for our model, we sample the output from the softmax layer, one event at a time, and feed each prediction back into the model to generate the next. During generation, we use *beam search* (Olive et al. 2011) to sample the most likely prediction at each step.

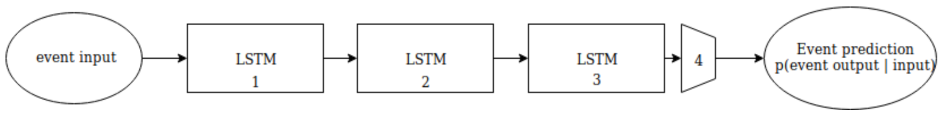


Figure 5.1: *Network architecture for generating MIDI music using a compressed encoding. Network components numbered for reference: 1,2,3 are LSTM cells, and 4 is a feed-forward layer with a softmax activation.*

[1]We note that such layers are also referred to as a Dense layer or Classification layer in certain cases.

5.2 CondRNN: Learning Compositional Style

The basic model outlined in the previous section is similar to prior work in relevant literature. While it can capture interesting musical structure on short timescales, it cannot learn discreet representations of different compositional styles, thus there is a clear gap in knowledge for a method which can model the dependencies between salient features that compose an abstract concept such as composing style.

To model these latent features, we aimed to demonstrate a simple approach that could prove our hypothesis by utilising composer labels in our training data. The hidden states of our model are powerful encodings of their inputs, and we could enforce structure on these hidden states by attaching a classifier to our model, as in 5.2, component 5. This would predict a softmax probability distribution over the composers, similar to component 4, but we can't easily optimize an objective function that combines the two outputs. A naive approach could take the cross-entropy loss of the composer classifier, and create an objective function that minimizes the mean of the two losses. However, this approach would mean that the model is likely to get stuck in local minima, i.e. the model's ability to predict composers very well could completely hinder it's ability to predict correct musical events.

To overcome this, we can consider transfer learning (Caruana 1997), and consider the two tasks each of realistic music generation, and conditional music generation, separately. By pre-training a model illustrated in 5.2 that only optimizes the event prediction loss, we can transfer the musically relevant knowledge that has been learned to classifying composers, and enforce structure on a model which already has some representation of musical concepts. We aim to do this by using a *growing weighted loss function*:

For a given sequence, our model produces two cross entropy losses, $l_e vents$ and $l_c omposers$. Given two corresponding weights, $w_e vents$, and $w_c omposers$, such that $w_e vents + w_c omposers = 1$ the overall loss function can be defined as:

$$(w_e vents * l_e vents) + (w_c omposers * l_c omposers)/2$$

We can use transfer learning by growing the weight $w_c omposers$ over the course of the model's training from a very small initialized value. Finally, we dub this new architecture *CondRNN*

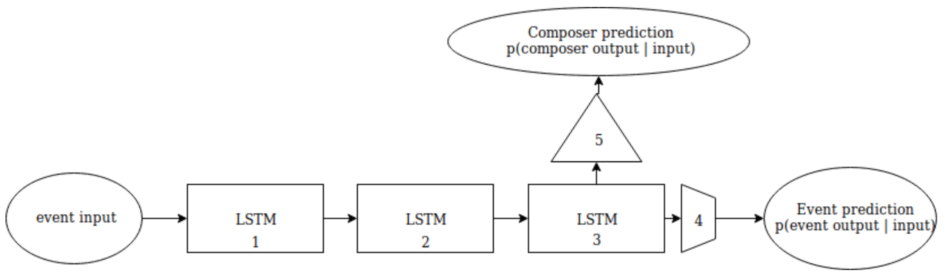


Figure 5.2: *Network architecture for imposing structure on internal LSTM states using a classifier. Network components numbered for reference: 1,2,3 are LSTM cells, 4 is a feed-forward layer with a softmax activation predicting the next event, and 5 is a feed-forward layer with a softmax activation predicting the composer.*

5.3 Data

Recurrent networks require a sequence of inputs, and are particularly sensitive to the representation of these inputs. In this section, we detail our dataset creation, data representation, and data augmentation methods.

5.3.1 Designing Data

We begin by following standard practice in machine learning and create subsets of our dataset, each 80% for training the model, 10% for validating, and 10% for testing. The training split of the dataset is intended purely for training the models. We include a validation dataset, which we use to evaluate models using quantitative metrics such as cross entropy loss. This allows us to compare different configurations of our models and test hyperparamter choices. The testing split of the dataset is for the purpose of evaluating a final model choice to determine it's ability to generalise. The essential part of subdividing our dataset is to ensure we can make empirically informed decisions about our model design choices. This is only made possible by being able to evaluate the model on completely unseen input data.

5.3.2 Music as Events

A straightforward data encoding would simply involve predicting MIDI events, one event at a time, on a uniform metric grid. Encodings for recurrent models are usually referred to as *vocabularies* that are represented with a *one-hot vector*. A one-hot vector represents each possible item in a vocabulary as a binary digit. Every step in a sequence is a single one-hot vector with only a single element being 1, and the rest being 0. A model which used such an encoding would essentially be predicting a single MIDI event at each fixed time-step. However, this representation would mean that the model would need to output a single event continually to hold notes, and the longer sequence length would reduce the model's ability to capture relevant long-term structure. The model would also be unable to capture dynamics of music, as all MIDI events would have a fixed velocity.

To better capture dynamics and expressiveness, we encode our data for our model by augmenting a one-hot MIDI vocabulary as in (Oore et al. 2018). The vocabulary is defined as:

- 128 NOTE-ON and 128 NOTE-OFF events, which each represent a single pitch on a piano. A NOTE-ON event begins playing a note, and a NOTE-OFF event "releases" a note.
- 100 TIME-SHIFT events, each event moves the time-step forward in the sequence. The time-shift events move time in increments of 10ms up to 1 second.
- 32 VELOCITY events. Each velocity event sets a specified velocity level to all subsequent notes until the next velocity event.

This representation discards a fixed meter, and instead fixes a *time step* at 10ms, allowing the model to "skip" forward in time using arbitrary TIME-SHIFT events. This compressed representation frees a model from having to learn to repeat note events for holding notes Sturm (2017), and is also crucial for polyphonic music [2], as the model is able to play multiple notes in a given time step before moving time forward. Thus, the model described in following sections will operate over such an encoding using a one-hot vector of length 388.

5.3.3 Composers as Numbers

We also use a one-hot encoding for representing our composers. Given a set of four composers: Bach, Debussy, Chopin, and Beethoven, we can consider them as a vector 4 bits long:

[2]A polyphony consists of more than one line of independent melody.

$[bach, debussy, chopin, beethoven]$. For any given sequence, our model will simultaneously predict a probability distribution over the next event, and also the probability distribution over our composer encoding, which might look like $[0.25, 0.25, 0.01, 0.49]$, for a trained model which is predicting a sequence composed by Beethoven. In this case, the corresponding ground truth would be $[0, 0, 0, 1]$.

5.3.4 Data Augmentation

We *augment* our data by creating transformed copies of our training examples to include in the training dataset. Increasing available examples improves our model's ability to generalize by forcing it to maintain high-level information. We augment our data following the method used in Oore et al. (2018). Firstly, we split our MIDI examples into 30 second clips, which are then transposed up and down all intervals up to major third[3], which results in 8 new transposed samples. We also stretch each example uniformly by -2.5% and -5%, and +2.5% and +5%, which provides 4 additional examples.

5.4 Summary

In this chapter we outlined our main contribution. We developed a novel transfer-learning based method for meaningfully structuring LSTM internal states according to compositional style. We discussed our dataset creation strategy, and proceed to detail our implementation process.

[3]A musical interval consisting of 3 semitones, the smallest musical interval on the piano.

6 Implementation

This chapter outlines the process of implementing our design, including dataset creation and pre-processing, model definitions, and training, evaluation and generation scripts, as well as the justification for various decisions made.

6.1 Background

Deep learning is becoming increasingly widely adopted, and this is reflected in the variety of tools that are available to aid development of large, scale-able deep learning models. In the following subsections we detail the tools we will be using.

6.1.1 Tensorflow

Tensorflow (Sinkinson 2017) is a machine learning framework that provides implementations for many common numerical operations in deep learning. With the advent of large-scale datasets, Tensorflow also supports large-scale data operations, and model training and visualisation, to name a few. We chose Tensorflow due to its widespread use, its low-level APIs that allow for custom neural architectures, and its support for training large networks quickly on GPUs. This made it straightforward to build upon existing work, while also having flexibility in how we implemented our models.

At a fundamental level, Tensorflow works by creating *dataflow graphs*, structures that describe operations as many interconnected nodes in a graph which data moves through. Each node in a graph represents some mathematical operation, and connections between nodes is the data that passed between them. These graphs are able to be saved and reloaded, and provide a de-coupled mechanism for creating numerical models.

6.1.2 Magenta

Google Magenta (Magenta) is an open-source research project which explores the use of machine learning in art, and much of the relevant prior work in this dissertation is a result of this project. The group maintains an open-source GitHub repository, which contains the code used by the researchers to develop their models. This dissertation is a fork of the Magenta GitHub, and builds upon much of the infrastructure provided to implement our models.

6.1.3 Cloud Training

Deep Learning models require intensive GPU hardware to efficiently, and the process of designing and evaluating these models is laborious. We used the Google Colaboratory (Google) environment as a test-bed for quickly implementing and validating different model designs. The

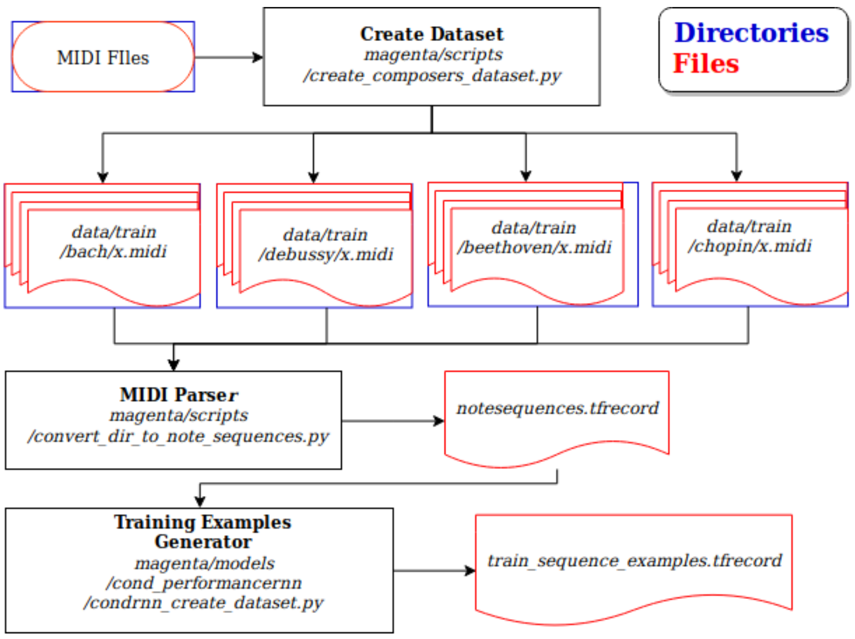


Figure 6.1: Overview of how the MAESTRO dataset is used to create training examples for our models, with references to specific points in the codebase.

Colaboratory environment provides a GPU-enabled cloud machine, and the availability of this tool helped our project tremendously. To train our final models, we used the Google Cloud Platform (GoogleCloud), which provides specialised GPU hardware and data-storage solutions online, and equipped us with the tools to scale our models.

6.2 Data

Our model requires a large amount of data, and many operations applied to the data before it can be used for training. 6.1 Provides an overview of the pipeline the dataset goes through in order to transform our data into training examples.

Each stage in the pipeline allows us to be flexible with the kinds of data we create, and what we use it for:

- **Create Dataset** The initial stage of data processing simply copies subsets of a whole dataset into directories, grouped by composer. The dataset optionally provides subsets for training, validation, or evaluation, and also allows each MIDI file to be split into a samples of specified length and quantity, which was incredibly useful as it allowed us to use a single script to generate different examples for all of the evaluations we performed.
- **MIDI Parser** This stage of the pipeline utilises the Magenta library to parse MIDI files into NoteSequences, which is the object class Magenta uses to represent MIDI data. The converted NoteSequences are stored in the *TFRecord* format, which is a record-oriented binary representation of data, and also plugs into the Tensorflow data processing API painlessly.
- **Training Examples Generator** The final stage of the data processing pipeline has been implemented manually. This stage takes NoteSequences and converts them to training examples using Magenta *Pipelines*. Pipelines allow operations that transform data to be chained together and are vital for data pre-processing and augmentation. Our data pipeline

appends composer labels to training examples, and also augments the data as in Section 5.3.4. Output files from this stage are ready to be used in a model.

6.3 Models

Our model definitions followed the format used in the Magenta repository. A model is simply a Tensorflow graph of operations that work on some input data, and the core element of our model definitions were a `get_build_graph_fn()` method. This function returns another function which, once executed, builds a Tensorflow graph that contains the model definition. All models share a definition of this function signature, which requires a run-time directory, and a "mode" which is one of 'train', 'eval', or 'generate'. We extend this definition to also require an instance of a `Config` class. The Config class captures important information about model hyperparameters and streamlines the processes of model training, and sharing parameters between models. These design choices were vital for reducing code repetition and also provided a consistent environment for evaluating our models in.

6.3.1 Model Training

We feed inputs to our models using an implementation of the Tensorflow `Dataset`, modified for use with our NoteSequence and composer label data. The Dataset provides a high-performance method for reading training data from TFRecord files. We trained our model in *batches*, which are fixed-sized portions of the training data and define the number of samples we propagate through our model at once. We also employed Teacher Forcing during training (Sargur) which uses the ground truth as the next input for the model during training, instead of the prediction from the model. This encourages our model to quickly learn correct statistical patterns in our sequences. For a domain such as music, these statistical patterns are vital to the model producing music with good repetition and structure.

Our models were trained on a single GPU with 16GB of RAM, for 22,000 steps. We used an LSTM cell size of 512 for all LSTM layers, and a fixed learning rate of 0.001. The baseline model used a batch size of 64. The models were trained with the RMSProp Optimizer (Hinton), and at each step we clipped the gradient updates with their L2 norm to prevent gradient explosion. (Pascanu et al. 2012).

6.3.2 CondRNN Training setup

We implemented the growing weighted loss using a polynomial function of the current training step. The weighted composer loss grew from 0 to 0.07 over 16,000 training steps. Not all music from a composer is homogeneous or able to be distinctly classified, and this required a large batch size of 128 to ensure unrepresentative examples in the dataset weren't causing un-optimal gradient steps. We conclude this section by providing the training graphs for the models in Appendix A.1.

6.4 Summary

We took advantage of the recent explosion in available Deep Learning tools and specialised hardware to create a novel recurrent network architecture from scratch. This section discussed key points of the implementation, and we look to place this in context with an evaluation.

7 Evaluation

Music is very subjective, and this makes precise evaluation of a model which generates music fundamentally difficult. There are no exact metrics for evaluating how realistic a piece of music is, or how accurately a piece of music embodies a certain compositional style, and standard measures for evaluating deep learning models can be misleading for evaluating a model with goals such as these. Theis et al. (2015) argue that the limitations of metrics for evaluating generative models means that models need to be evaluated directly, with respect to their intended application. Given that our intended application is to create realistic music, and to learn compositional style, it makes sense to evaluate these directly. We aimed to evaluate our models both qualitatively and quantitatively as thoroughly as we could. In the following sections we provide an overview of our evaluation strategy, the methods we used and the results they yielded.

7.1 Evaluation Strategy

In order to measure the performance of our model, we needed a meaningful reference point to compare our model to. For the remainder of this chapter, we will refer to a *baseline* model, and direct the user to Section 6.3.1 for precise definitions of our baseline model. This allows us to determine whether the novel method implemented in Section.5.2 is improving our approach to the problem. We conducted both qualitative and quantitative evaluations in this project to measure how well we achieved our research hypothesis, and begin by providing details on our quantitative evaluations.

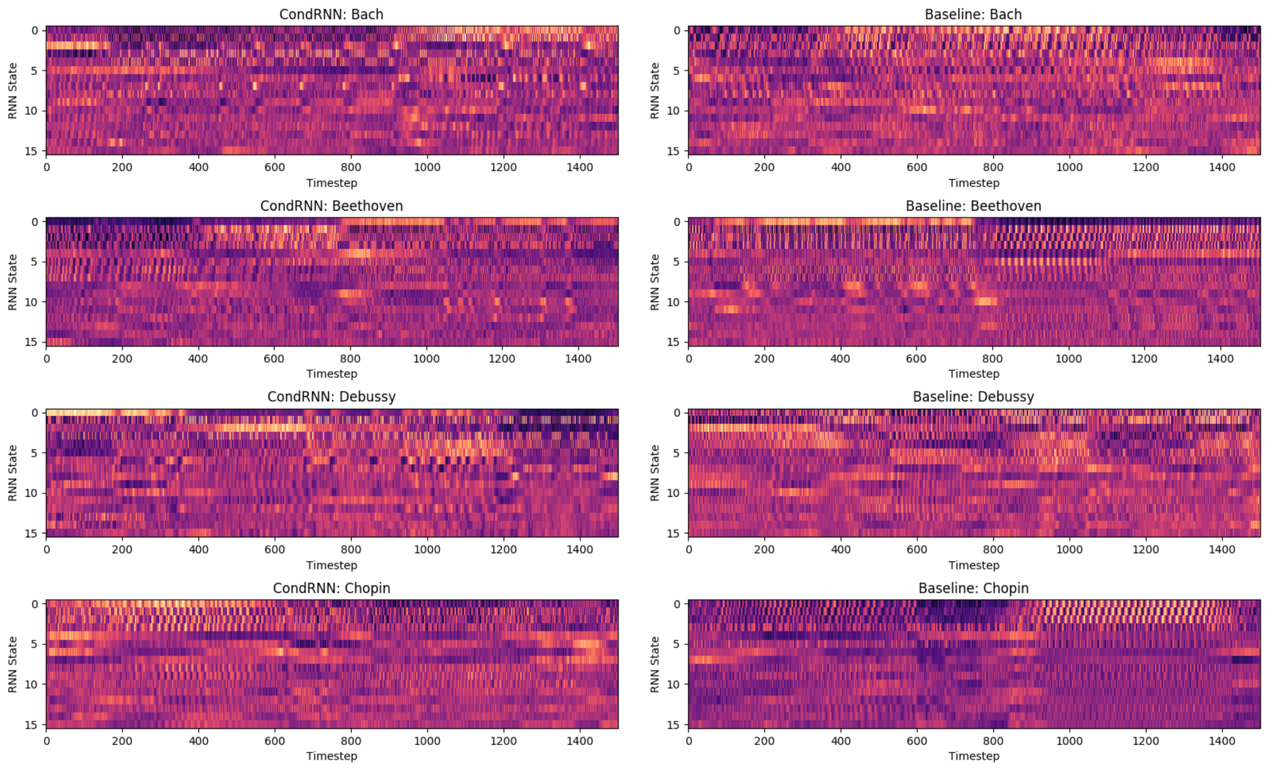
7.2 Quantitative Evaluations

Despite objective evaluations of the kinds of generative models we present in this dissertation, we attempt to provide comparisons over our baseline across several quantitative measures.

Table 7.1 contains the average cross-entropy loss of the two models. Both models were evaluated on the unseen, held-out portion of the dataset. We note that our model performs better than the baseline, but both losses are relatively high and indicate *over fitting*, which is an issue with models which have only learned the statistical attributes of the data they have been trained on, and are unable to generalise. We might attribute the better performance from CondRNN is due to the extra information utilised in the composer labels, allowing the model to better generalise. We could reduce overfitting in this case by providing a larger dataset and a longer training time.

Model	Cross Entropy Loss
Baseline	4.294
CondRNN	3.963

***Table** 7.1: Model performance on unseen data.*



(a) States for music generated from different composers, CondRNN *(b) States for music generated from different composers, baseline model*

Figure 7.1: *RNN states, projected to 16 dimensions, for 15 seconds of generated music. A timestep corresponds to a single RNN state, and each second is 100 RNN states. Each model was primed with the same 15-second sample for the same composer, from the held-out dataset.*

The remainder of this section will aim to address our research hypothesis by performing experiments directly on the *hidden states* of our models.

7.2.1 Visualising Structure in Generated Music

Repetition is a fundamental aspect of music, and repeated motifs or melodic lines serve to legitimise a piece of music. In this evaluation, we investigate our model's ability to capture structure in generated sequences by repeating relevant musical sections. The hidden of a model are representative of the information stored in the model, and our assumption is that this information corresponds to musically relevant temporal structure (Liang et al. 2017). Thus, for any given sequence that a model generates, we can capture the final hidden states of our model, and attempt to determine if there is some temporal structure in the states themselves.

Figure 7.1 shows the RNN states for the baseline and CondRNN generating the same 15-second pieces of music. These visualisations demonstrate repeating structures in both models as similarly-coloured areas of the visualisation with regular intervals. We also calculated the *variance* across all states for these sequences, and Table 7.2 shows that the baseline deviates more from a representative mean than CondRNN.

	Variance over RNN states per composer			
	Bach	Debussy	Chopin	Beethoven
Baseline	0.179	0.185	0.175	0.172
CondRNN	0.159	0.150	0.152	0.175

Table 7.2: *Variance over RNN states for generated sequences.*

7.2.2 Visualising Structure in our Model

Our goal was to impose constraints on the hidden states of our model, such that the model is able to distinctly represent the different compositional styles. In section DESIGN Sec we demonstrated a novel method which optimizes an objective function that aims to achieve this discreet representation. We evaluate in this section the extent to which our model has disentangled its hidden states.

Similarly to the previous evaluation, in this experiment we capture the hidden states for generated music sequences. We randomly sampled 5 second sequences from songs in the held-out dataset, and sub-divided these samples per composer. Overall, we used 800 5-second samples from each composer to prime each model, and each 5-second sample resulted in 500 RNN hidden states. We took the *median* of these 500 hidden states to produce a single RNN state for each song.

The t-SNE dimensionality reduction algorithm (Laurens van der Maaten and Hinton 2008) is widely used for projecting high-dimensional spaces into lower dimensions for visualisation. We projected 800 RNN vectors for each composer from 512 dimensions to 2, for each model. Visual inspection in Figure 7.2 shows a clear structure in CondRNN, whereas the baseline model shows no grouping of similar composers whatsoever.

We produced two metrics to evaluate precisely the extent a model is able to discriminate between different composers:

- **Mean intra-composer distance**
  We define the Euclidean Distance Anton and Rorres (1994) between two vectors Anton and Rorres (1994), x, y, as

$$dist(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

We extend this definition to operate on two matrices Anton and Rorres (1994), X, Y, of shapes AB and CD, respectively, as a *distance matrix* of shape AC

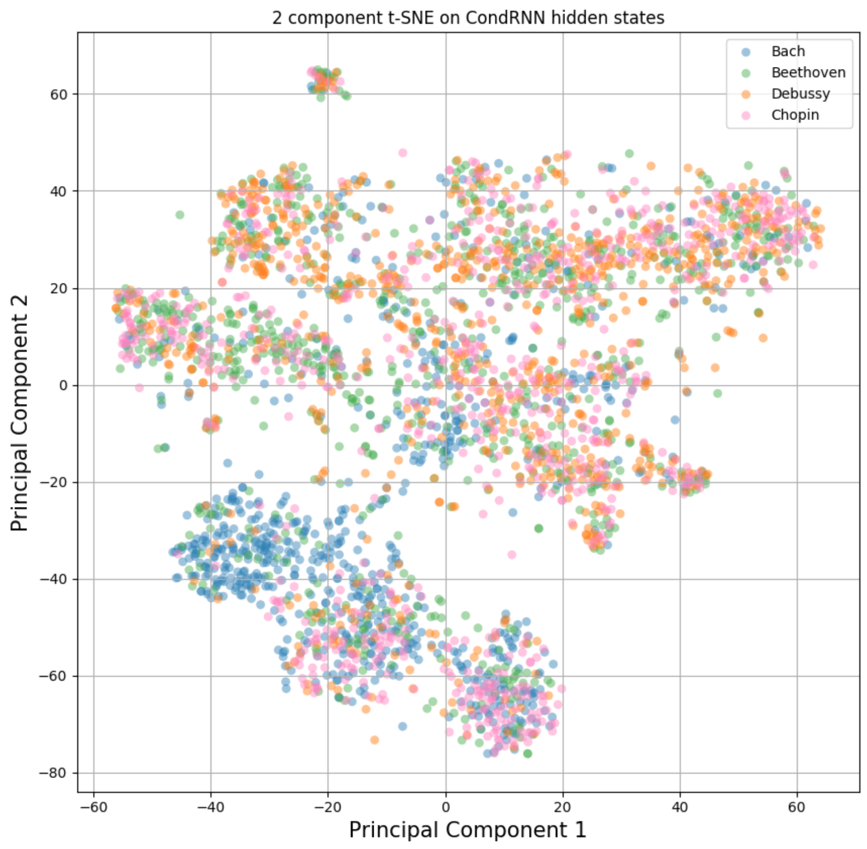
$$dist\_mat(X, Y) = Z_{AC} \text{ , where } Z_{i_j} = dist(X_i, Y_j), \text{ and } Z_{i_j} = 0 \text{ if } i == j \text{ and } X_i == Y_j$$

The intra-composer distance is defined as follows. Given a matrix X_c of n RNN states for a composer, $c \in \{Bach, Debussy, Beethoven, Chopin\}$, we have

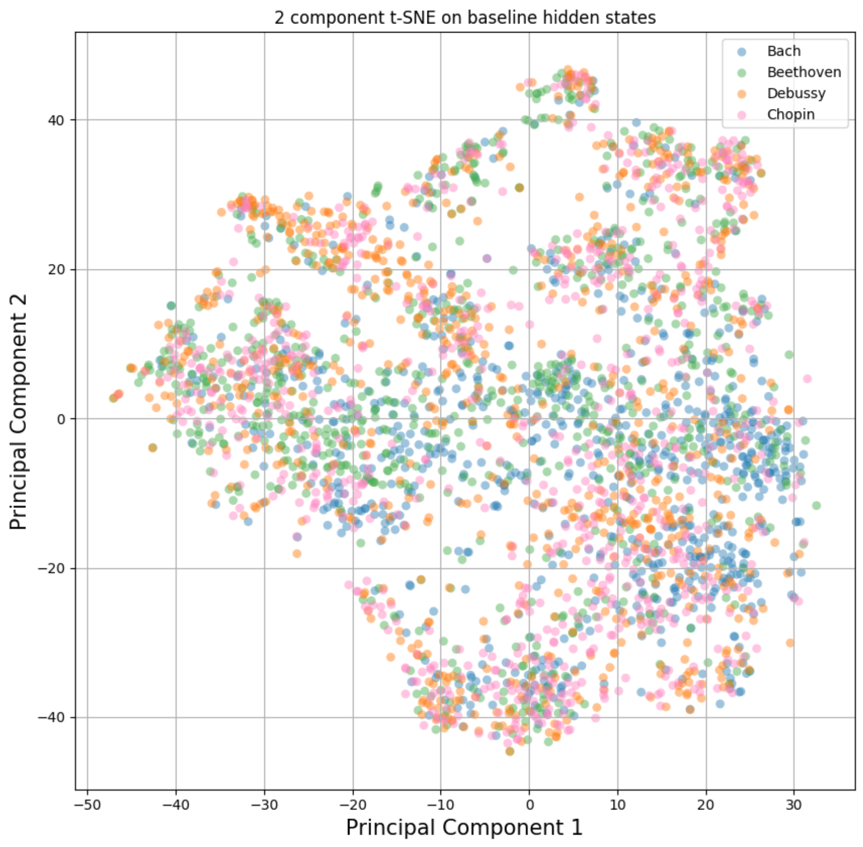
$$intra\_composer\_dist(X_c) = \frac{1}{n * n}\sum_{i=1}^{n}\sum_{j=1}^{n} dist\_mat(X_c, X_c)_{i_j}$$

Furthermore, we can summarise this into a single metric for a model, for a matrix X of N points, M-dimensional RNN states for C composers, with shape CNM and for all $c \in C = \{Bach, Debussy, Beethoven, Chopin\}$

$$mean\_intra\_composer\_dist(X) = \frac{1}{4}\sum_{i=1}^{n} dist\_mat(X_c, X_c)$$



(a) States for music generated from different composers, CondRNN



(b) States for music generated from different composers, baseline model

Figure 7.2: *RNN states, projected to 16 dimensions, for 15 seconds of generated music. A timestep corresponds to a single RNN state, and each second is 100 RNN states. Each model was primed with the same 15-second sample for the same composer, from the held-out dataset.*

- **Inter-composer distance**
  Given matrices X_{c_1}, X_{c_2}, each AM, BM of shape A and B points, M-dimensional RNN states respectively, for c_1, c_2 in $\{Bach, Debussy, Beethoven, Chopin\}$, we define the inter-composer distance as

$$inter\_composer\_dist(X_{c_1}, X_{c_2}) = \frac{1}{N * M} \sum_{i=1}^{N} \sum_{j=1}^{M} dist\_mat(X_{c_1}, X_{c_1})_{ij}$$

  We can similarly summarise this into a single metric, for a matrix X of N points, M-dimensional RNN states for C composers, with shape CNM, and for all $c_1, c_2 \in C = \{Bach, Debussy, Beethoven, Chopin\}$, we have:

$$mean\_inter\_composer\_dist(X) = \frac{1}{16} \sum_{i=1}^{4} \sum_{i=1}^{4} inter\_composer\_dist(X_{c_i}, X_{c_j})_{ij}$$

- **Inter-intra composer distances difference**
  We can see that a sensible objective for our model would be to maximise the following, for a matrix X of N points, M-dimensional RNN states for C composers, with shape CNM, and for $C = \{Bach, Debussy, Beethoven, Chopin\}$,

$$|mean\_inter\_composer\_dist(X) - mean\_intra\_composer\_dist(X)|$$

These metrics allow us to determine how the hidden states of our model are structured with respect to the different composers in the dataset. Table 7.3 displays these metrics for the two models to allow for comparison.

	Mean Intra-Composer Distances	Mean Inter-Composer Distances	Inter-Intra Composer Distances Difference
Baseline	0.562 +- 0.36	0.659 +- 0.24	0.097 +- 0.07
CondRNN	0.642 +- 0.385	0.803 +- 0.233	0.161 +- 0.09

Table 7.3: Metrics on model performance for structuring latent states.

7.2.3 Summary

Overall, our model demonstrates a clear improvement when evaluated with quantitative metrics against a baseline. We demonstrated that CondRNN is able to structure its latent space meaningfully, with sequences by the same composer clustered together and distant from sequences from other composers. We also demonstrated that the model performed better on unseen data, which indicates that learning to disentangle the internal states of a model can help it learn higher-level information and better generalize.

7.3 Qualitative Evaluations

The performance of the model on unseen data paints an incomplete picture. Music is inherently a human experience, and human understanding is the ultimate expression of music. The following subsections will detail our qualitative evaluation approach which aimed to answer questions about the most relevant aspects of our model's performance.

7.3.1 Duo Trio Test

The Duo-Trio Test (Stone and Sidel 2004) is a technique used in sensory analysis for discriminating two items based on human perception of them. The test uses groups of human panellists to determine if there are detectable differences among two items. A typical setup involves each subject being presented with an identified reference or *anchor* sample, and two coded samples, one which is the same as the identified reference, and the participant must choose which of the two is distinct from the reference. The test was particularly relevant for this evaluation since we often wish to direct participants in the test to discriminate based on specific attributes or qualities of a sample of music, e.g. realism, or compositional style. However, describing these concepts precisely is difficult, and "talking about music is like dancing about architecture" (Mottier 2009). Furthermore, a certain level of musical knowledge and prior experience is assumed, and listeners may not have a classical background, or even have heard much classical piano music prior to the test. By selecting an anchor sample which carried the attributes and qualities which we hoped would act a a reference point with relevant features for users to discriminate with.

The principle aim of a duo-trio test is to reject a *null hypothesis* Draper (2011), which states that *there are no detectable differences between the two items*. In order to do this, evidence is collected from the test and a statistical test is performed on the evidence. If there is sufficient evidence from these results, then we reject the null hypothesis for the *alternative hypothesis*, which states that *there are detectable differences between the two items*.

7.3.2 Experimental Setup

It's necessary to be exact about what evidence is sufficient to reject a null hypothesis. A statistical test provides a mechanism for making a quantitative decision about evidence based on underlying statistical principles, and usually determines a *significance level* about that decision. A significance level is the probability of rejecting the null hypothesis given that it is indeed true.

We conducted three evaluations with 8 participants. For this number of assessors, we require 7 participants to score correct answers for a significance level of 5%, and 8 participants to score correctly for a significance level of 1%. For each of our experiments, users were presented with three samples of music, each 10 seconds long. In each set of three samples, one was an anchor, and the other two were samples to discriminate between. We selected a sample of 10 seconds after an initial pilot study which tested subject's discriminating accuracy on different lengths of sample.

For a given experiment, users are presented with three samples which we will refer to as A, X, and Y. For all further references to this format, the sample labelled A will be the anchor sample, and in experiments which evaluate the model presented in this project, the model developed will be labelled X and the baseline model labelled Y.

Our experiments were conducted in Google Colaboratory, (Google), which is a free, online, cloud-based Python notebook execution environment (Jupyter), and Google Forms (Google 2019), which is an online tool for collecting information from users with surveys. Code is executed in these notebooks as cells, and each cell displays the output of the code inside it individually. Each participant was provided with a unique notebook and an online copy of the survey, which contained instructions on how to run the experiment. We include the notebook and the survey in Appendix A.2 for the reader. Users would run a cell which had its code hidden, and would listen to three samples of music, one being indicated as the anchor, and would select the sample they discriminate in the respective question on the survey. Subjects took part in the experiment remotely.

Even with a reference point, a certain level of experience must be assumed about the participant. We therefore asked subjects to detail relevant prior experience before the experiments began.

7.3.3 Participant Experience

We briefly note on the participant experience, and provide data in Appendix A.2 for the reader along with Ethical Approval. Only 1 participant responded that they couldn't read sheet music, and 75% of participants had at least 3 years of prior formal musical training. Given that we were generating western classical music, we felt it was particularly relevant that participants had some familiarity with western music, and only 2 participants didn't use English as a first language. We detail our remaining experiments below, but recommend the reader to take these preliminary results with a grain of salt, as the survey suffered from a very small sample size, and participants were likely to over-estimate their abilities.

7.3.4 A Turing Test of Sorts

Are humans able to perceive the difference between music generated by our model, and a baseline, when asked to discriminate based on realism?

In this experiment, the A sample is taken from a song which is randomly selected from the held-out portion of the dataset, and also randomly selected from some point within that song. The X and Y sample are generated by priming each model with the prior 10 seconds of the sample of the anchor. We repeated this experiment 8 times per subject, to ensure there was a random assignment of the different possible orderings of the samples.

We aimed to evaluate the realism of the music generated, and asked subjects to select the sample which sounded least like it was made by a human. Crucially, subjects were told that the labelled anchor was a human recording. The null hypothesis is that subjects discriminate against CondRNN more frequently than the baseline. The alternative hypothesis is that subjects discriminate against the baseline model more frequently than CondRNN.

We show the results in 7.3. The mean of the frequencies was $\frac{4}{8}$ for each model. Given that $\frac{4}{8} < \frac{8}{8}$, we can reject our alternative hypothesis with a significance level of 99%.

7.3.5 Can humans detect compositional style

Are humans able to perceive the difference between music created by different composers?

In this experiment, the A sample is taken from a song which is randomly selected from the set of songs by a particular composer, from the held-out portion of the dataset, and also randomly selected from some point within that song. The X sample in this experiment is a sample similarly chosen from the same composer, and the Y sample is chosen from the set of songs from a different composer. We repeated this experiment 6 times per subject, to ensure there was a random assignment of the different possible orderings of the samples.

We aimed to evaluate whether humans were able to distinguish between samples from different composers, and asked subjects to select the sample which sounded least like it was made by the anchor composer. The null hypothesis is that subjects discriminate against the incorrect composer more frequently than the correct composer. The alternative hypothesis is that subjects discriminate against the correct composer more frequently than the incorrect composer.

We show the results in 7.4. The mean of the frequencies was $\frac{5}{8}$ for each model. Given that $\frac{5}{8} < \frac{8}{8}$, we can reject our alternative hypothesis with a significance level of 99%.

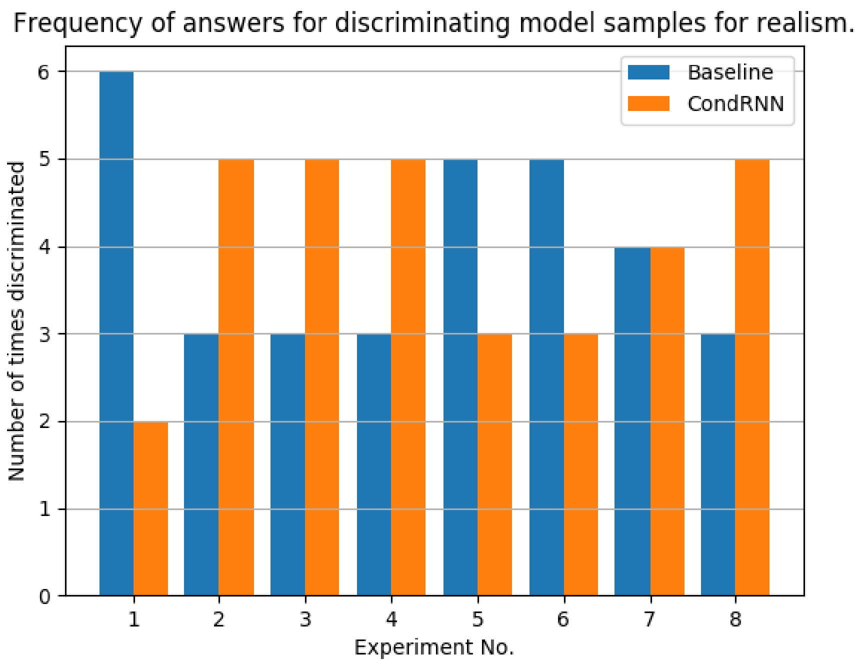


Figure 7.3: Results from Experiment 1

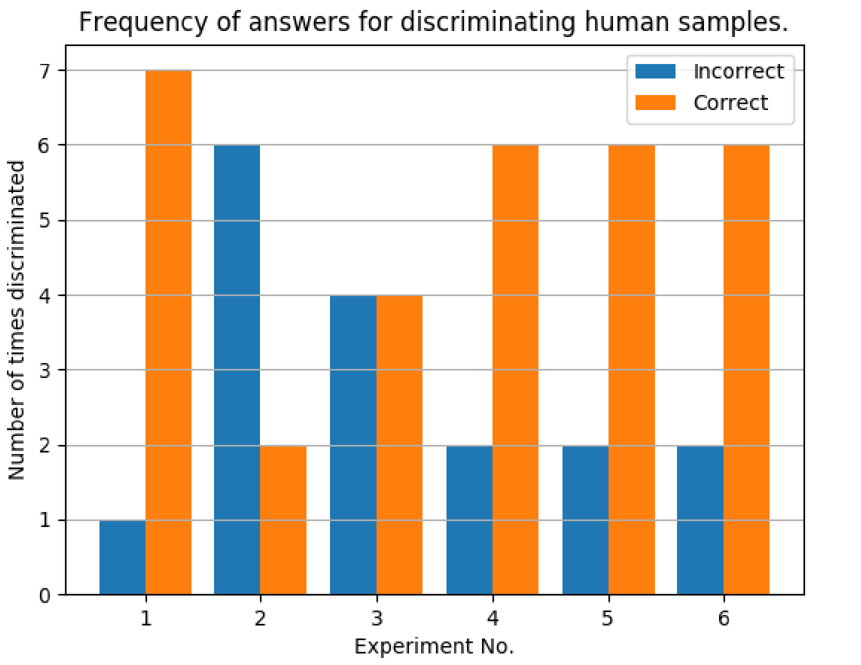


Figure 7.4: Results from Experiment 2

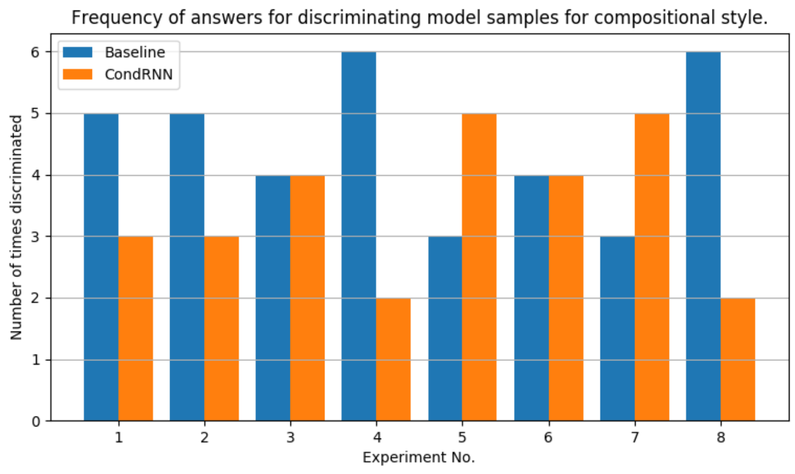


Figure 7.5: Results from Experiment 2

7.3.6 Not a Test of Skill

Are humans able to perceive the difference between music generated by our model, and a baseline, when the models are primed with a sample from a specific composer?

In this experiment, the A sample is taken from a song which is randomly selected from the held-out portion of the dataset, and also randomly selected from some point within that song. The X and Y sample are generated by priming each model with the prior 10 seconds of the sample of the anchor. All samples for a single experiment were from the same composer, and we aimed to determine whether humans could detect the difference in the baseline not being able to specifically capture compositional style.

We aimed to evaluate whether humans were able to distinguish between samples from CondRNN and a baseline. We asked subjects to select the sample which sounded least like the anchor sample. The null hypothesis is that subjects discriminate against CondRNN more frequently than the baseline. The alternative hypothesis is that subjects discriminate against the baseline model more frequently than CondRNN.

We show the results in 7.5. The mean of the frequencies was $\frac{5}{8}$ for each model. Given that $\frac{5}{8} < \frac{8}{8}$, we can reject our alternative hypothesis with a significance level of 99%.

7.3.7 Summary

Overall, we reflect on our qualitative evaluations and can see that our methods proved results which were inconclusive. The test we used is unsuitable for small sample sizes, and requires a very high margin for two items being distinct in order to be statistically significant. Music is entirely subjective, and such subjectivity leaves a wide margin for error. We believe a more conclusive evaluation could benefit from a much larger sample size. In addition, we might improve the consistency of our results by testing for similarity, rather than discriminating against it, or even providing a mock-test, as informal feedback from participants indicated that the test setup was confusing.

7.4 Discussion and Results

We aimed to evaluate our model against several criteria with the goal of proving our research hypothesis; to disentangle the latent space of a model in order to conditionally generate music based on composer style. Standard quantitative evaluations used in Deep Learning literature were not insightful, however, we designed evaluations and metrics that allowed us to test our models directly with respect to our hypothesis, and we found significant evidence which proves our hypothesis. Our qualitative evaluations suffered from a small sample size, and the inherently subjective nature of the topic, but could be improved by reducing the scope for confusion in the tests and a much larger sample size.

As previously stated in this dissertation, we believe that the most complete way to experience music is by listening to it, and provide a directory of samples for the reader in A.3. The samples includes human music, and also generated pieces from both the baseline and CondRNN which have been primed on the human music.

8 Conclusion

We have considered several approaches to the problem of generating music conditionally, and demonstrate a feasible method to conditionally generate symbolic MIDI music, conditioned on a composer style. We describe an effective model for conditionally generate music, and provide a detailed and thorough evaluation which directly tests our model against our research hypothesis.

We demonstrate a system that can create classical piano music, but also capture the complex temporal dependencies that occur in music at various levels. This represents an ability of the model to represent such dependencies in an incredibly efficient manner, and provides a proven method to base further work on.

A Appendices

A.1 Training Graphs

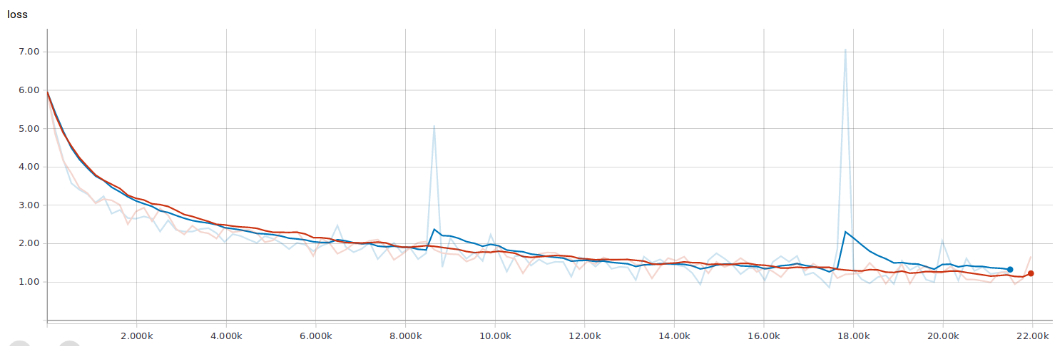


Figure A.1: *Cross-entropy loss for the baseline and CondRNN over 22,000 steps. The CondRNN is in blue.*

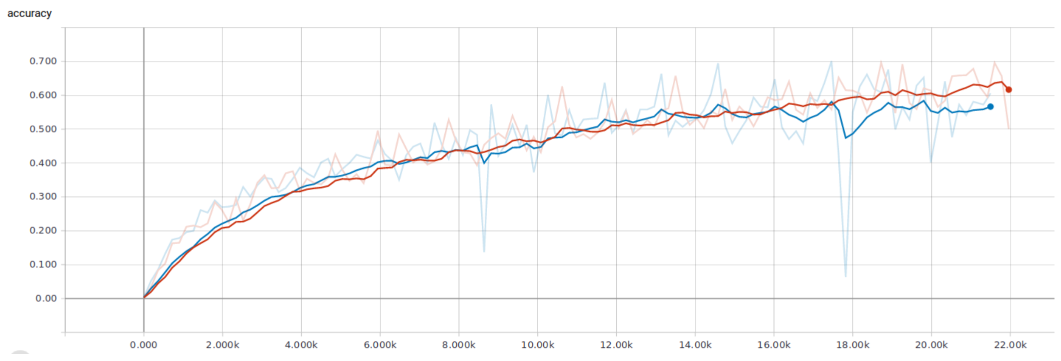


Figure A.2: *Accuracy for the baseline and CondRNN over 22,000 steps. The CondRNN is in blue.*

A.2 Surveys and Notebook

Exploring Artificial Intelligence and Music

CS4, 2018/2019
Salman Mohammadi
salman.mohammadi@outlook.com

The aim of this experiment is to evaluate a novel deep learning method for conditionally generating music, i.e. trying to get a computer to compose music which has a certain style or attribute. It's difficult to evaluate how good the music is unless we ask humans to listen to it, which is why we require human involvement. In the experiment, you will be asked to answer a survey which outlines your musical background, and then listen to samples of music and answer some mulitple-choice questions. All data collected will be anonymous, and you are welcome to withdraw from the test at any point. Please feel free to ask any questions now before the start of the experiment using the contact details provided.

This is not a test of your ability! We're evaluating the model, not how well you are able to answer the questions.

The evaluation should take no more than 10 minutes.

If you consent to taking part in this experiment, enter your provided participant number below, press next and follow the instructions provided.

* Required

1. **Participant number** *

Musical Background

This section will aim to assess your prior musical experience and training. If you're unsure about any of the questions, be conservative with your estimates. As stated previously, this is not a test of your ability.

2. **How many years of formal musical training do you have? (0 if none)** *

3. **Can you read sheet music?** *

   Mark only one oval.

   - Yes
   - No

4. **Is English your first langauge?** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No
   - ◯ I don't know

5. **Did you grow up listening to mostly English music?** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No
   - ◯ I don't know

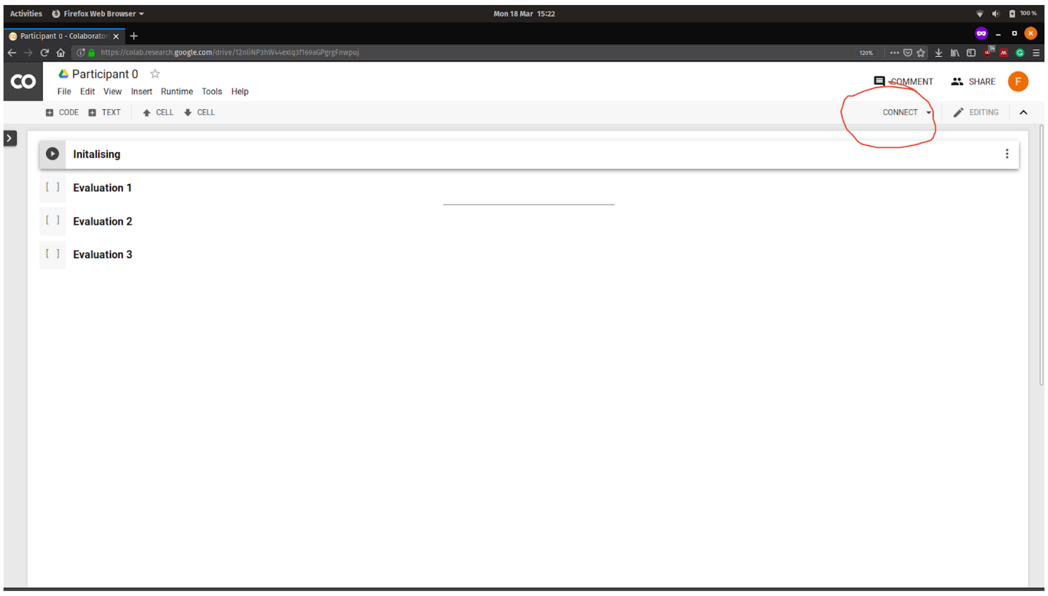
6. **Do you regularly listen to and enjoy classical music?** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No

7. **Tick all of the following composers which you would be able to distinctly recognise musical pieces by.**
   Check all that apply.
   - ☐ Chopin
   - ☐ Bach
   - ☐ Beethoven
   - ☐ Debussy

Experiment Setup

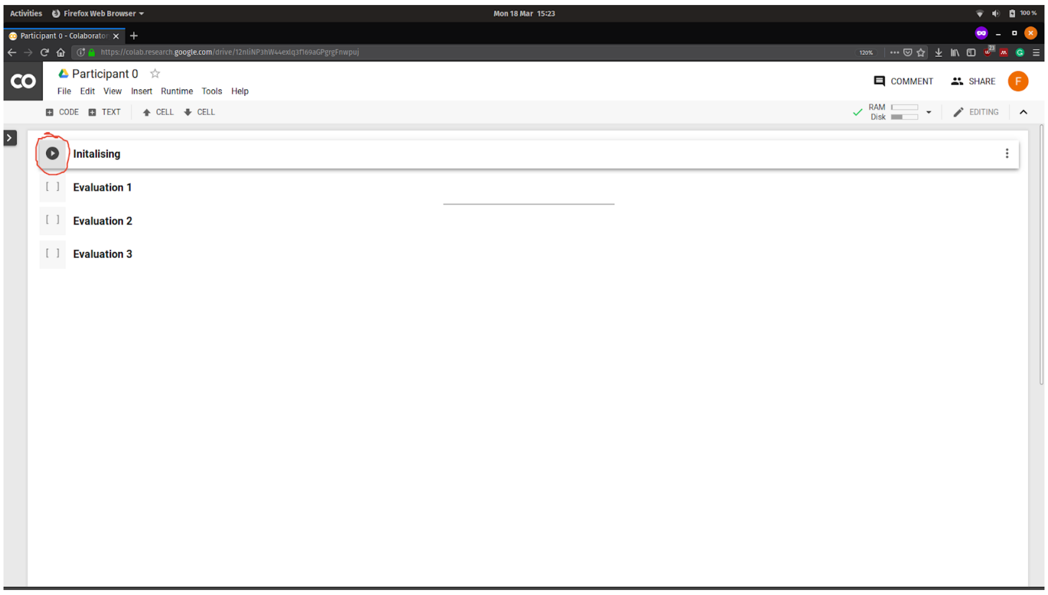
At this point you should open the link I sent along with this form. You'll need a google account to be able to run the experiments, but rest assured all data is anonymous and you only need to log into be able to use google's free online notebook platform.

Click "Connect" in the top right



Once it's done (it may take a while), you should see the image below. Follow the next step.

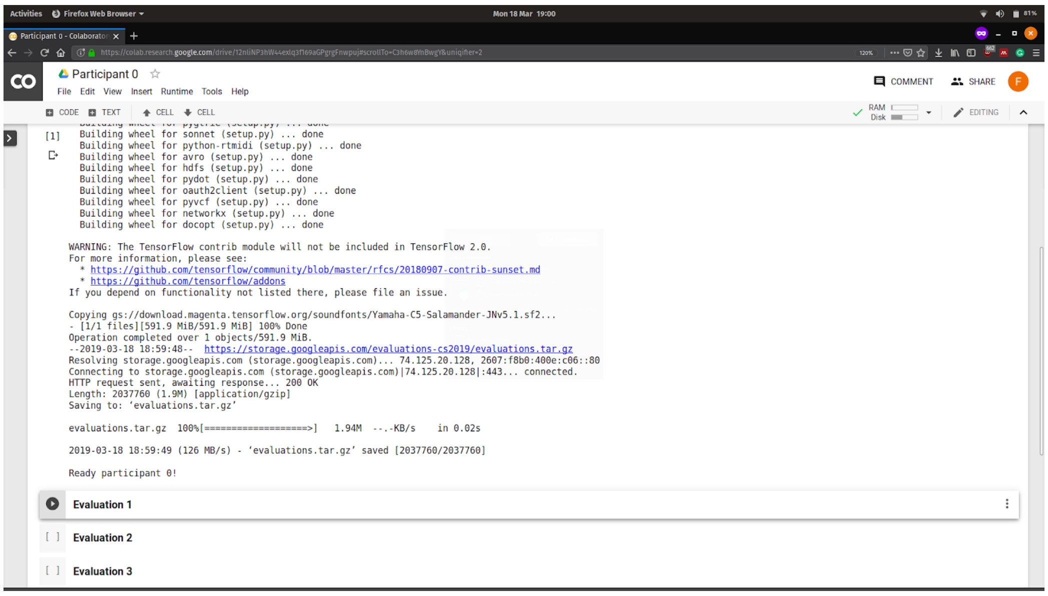
Hit the "Play" arrow shaped button highlighted.



You might get a warning about the notebook not being authored by Google. Hit "Run Anyway" to proceed. If another pop-up appears saying "Reset all runtimes", hit Yes.

I'm happy to share full source code after the experiment.

Once it's done running (it may take a minute or two), you should see the following screen.



```
[1]   Building wheel for sonnet (setup.py) ... done
      Building wheel for python-rtmidi (setup.py) ... done
      Building wheel for avro (setup.py) ... done
      Building wheel for hdfs (setup.py) ... done
      Building wheel for pydot (setup.py) ... done
      Building wheel for oauth2client (setup.py) ... done
      Building wheel for pyvcf (setup.py) ... done
      Building wheel for networkx (setup.py) ... done
      Building wheel for docopt (setup.py) ... done

    WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
    For more information, please see:
      * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
      * https://github.com/tensorflow/addons
    If you depend on functionality not listed there, please file an issue.

    Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
    - [1/1 files][591.9 MiB/591.9 MiB] 100% Done
    Operation completed over 1 objects/591.9 MiB.
    --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: ‘evaluations.tar.gz’

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - ‘evaluations.tar.gz’ saved [2037760/2037760]

    Ready participant 0!
```

You can now proceed to the first evaluation, press next below to begin.

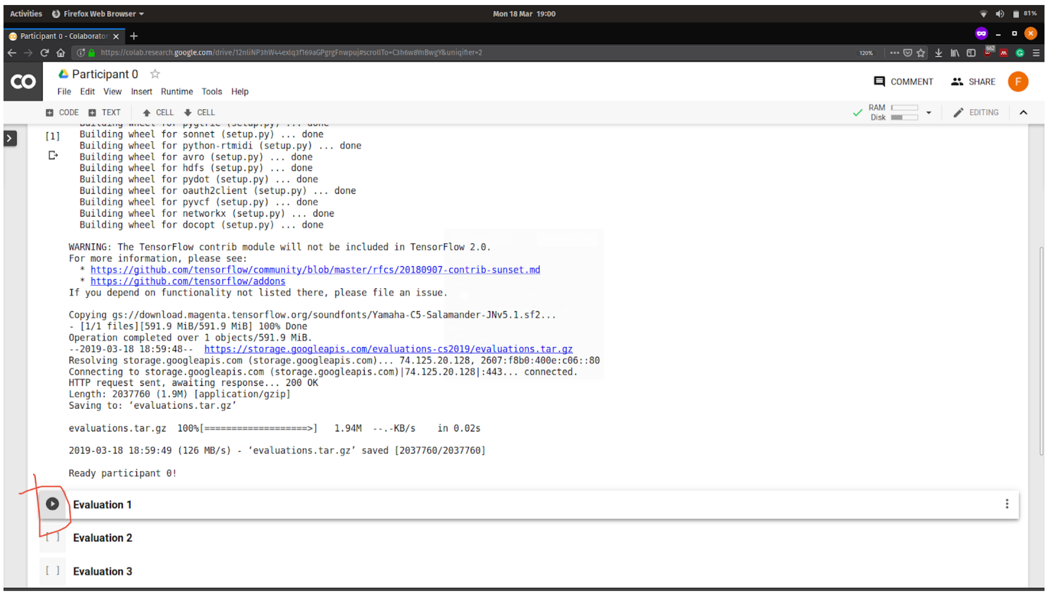
Evaluation 1

You will be given 8 experiments with three samples of music each, each sample 10 seconds long. In each experiment, you will be given an "anchor" sample, which is a human. You must select one sample from the remaining two that sounds LEAST like it was made by a human.

The instructions for this evaluation are as follows:

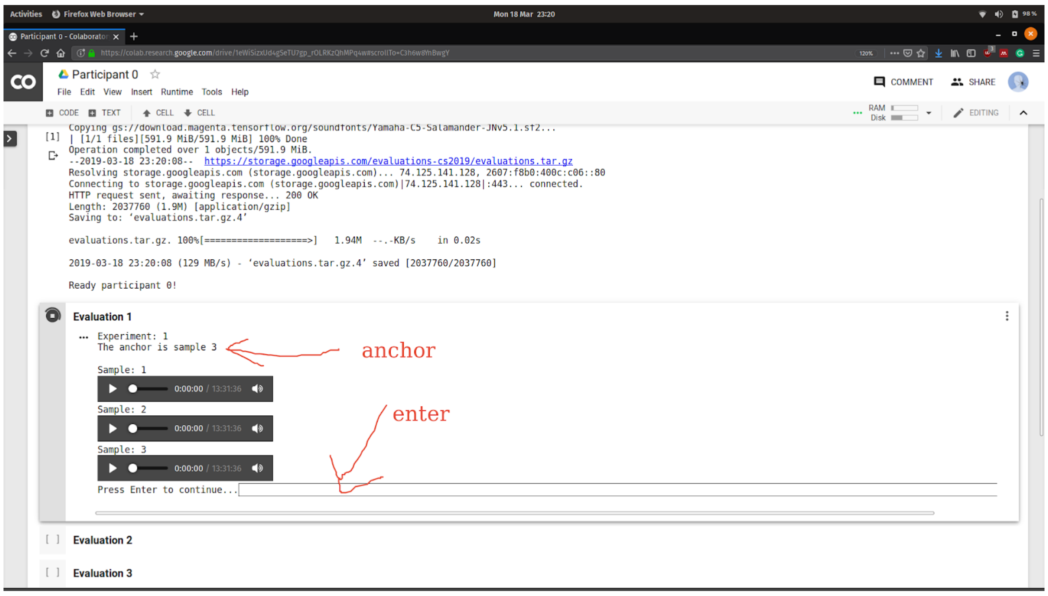
To run the evaluation

Hit the "Play" arrow shaped button highlighted.



```
[1]   Building wheel for sonnet (setup.py) ... done
      Building wheel for python-rtmidi (setup.py) ... done
      Building wheel for avro (setup.py) ... done
      Building wheel for hdfs (setup.py) ... done
      Building wheel for pydot (setup.py) ... done
      Building wheel for oauth2client (setup.py) ... done
      Building wheel for pyvcf (setup.py) ... done
      Building wheel for networkx (setup.py) ... done
      Building wheel for docopt (setup.py) ... done

    WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
    For more information, please see:
      * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
      * https://github.com/tensorflow/addons
    If you depend on functionality not listed there, please file an issue.

    Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
    - [1/1 files][591.9 MiB/591.9 MiB] 100% Done
    Operation completed over 1 objects/591.9 MiB.
    --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: ‘evaluations.tar.gz’

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - ‘evaluations.tar.gz’ saved [2037760/2037760]

    Ready participant 0!
```

Press the play button on each of the samples. You can listen to each one as many times as you like.


Activities
Firefox Web Browser
Mon 18 Mar 23:20
Participant 0 - Colaborator
https://colab.research.google.com/drive/1eWiSizxUd4gSeTU7gp_rOLRKzQhMPq4w#scrollTo=C3h6w8YnBwgY
Participant 0
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
COMMENT SHARE
RAM Disk
EDITING
anchor
enter

```
Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
| [1/1 files][591.9 MiB/591.9 MiB] 100% Done
Operation completed over 1 objects/591.9 MiB.
--2019-03-18 23:20:08--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.141.128, 2607:f8b0:400c:c06::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.141.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2037760 (1.9M) [application/gzip]
Saving to: ‘evaluations.tar.gz.4’

evaluations.tar.gz. 100%[===================>]   1.94M  --.-KB/s    in 0.02s

2019-03-18 23:20:08 (129 MB/s) - ‘evaluations.tar.gz.4’ saved [2037760/2037760]

Ready participant 0!
```

Evaluation 1

```
Experiment: 1
The anchor is sample 3

Sample: 1
0:00:00 / 13:31:36
Sample: 2
0:00:00 / 13:31:36
Sample: 3
0:00:00 / 13:31:36
Press Enter to continue...
```

Evaluation 2

Evaluation 3

In the example above, the user must select one from the remaining samples (e.g. 1 or 2) that sounds least similar to sample 3. Follow the instructions and use your own link to complete the evaluation.

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

8. **1 - Experiment 1** *

Mark only one oval.

◯ 1

◯ 2

◯ 3

Once you're done, fill out the question for the corresponding experiment, and click in the box shown and press Enter.

The next set of samples for the next experiment will now be available to listen to. After answering each experiment, click in the box and press enter to proceed to the next question.

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

9. **1 - Experiment 2** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

10. **1 - Experiment 3** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

11. **1 - Experiment 4** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

12. **1 - Experiment 5** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

13. **1 - Experiment 6** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

14. **1 - Experiment 7** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

15. **1 - Experiment 8** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

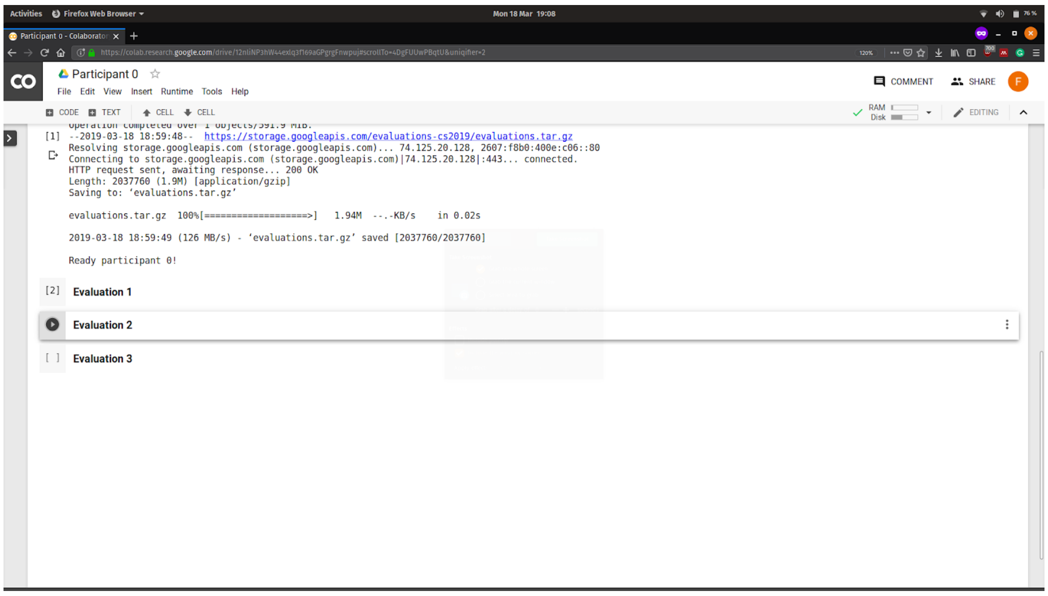
Evaluation 2

You will be given 6 experiments with three samples of music each, each sample 10 seconds long and created by a human. In each experiment, you will be given an "anchor" sample. You must select one sample from the remaining two that sounds LEAST like it came from the same composer as the anchor.

To run the evaluation:

To begin this experiment, you should ensure that you have answered all previous questions from the last section and you've pressed "Enter" on the final experiment from evaluation 1.

Hit "Play" to begin this evaluation.


Activities
Firefox Web Browser
Mon 18 Mar 19:08
Participant 0 - Colaborator
https://colab.research.google.com/drive/12nliNP3hW44exlq3f169aGPgrgFnwpuj#scrollTo=4DgFUUwPBqtU&uniqifier=2
Participant 0
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
COMMENT SHARE
RAM Disk
EDITING

```
Operation completed over 1 objects/591.9 MiB.
[1] --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: ‘evaluations.tar.gz’

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - ‘evaluations.tar.gz’ saved [2037760/2037760]

    Ready participant 0!
```

[2] Evaluation 1

Evaluation 2

[ ] Evaluation 3

The instructions are exactly the same as last time: listen to the samples, answer the corresponding question, and hit Enter to proceed.

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

16. **2 - Experiment 1** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

17. **2 - Experiment 2** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

18. **2 - Experiment 3** *
Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

19. **2 - Experiment 4** *
Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

20. **2 - Experiment 5** *
Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

21. **2 - Experiment 6** *
Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Evaluation 3

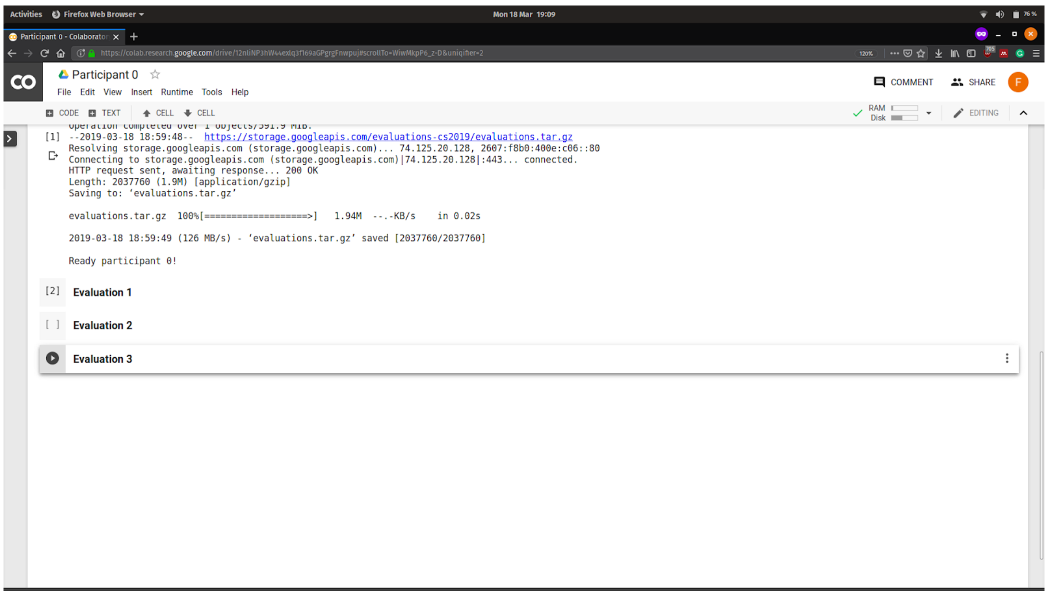
You will be given 8 experiments with three samples of music each, each sample 10 seconds long. In each experiment, you will be given an "anchor" sample. You must select one sample from the remaining two that sounds LEAST like the anchor sample.

To run the evaluation:

To begin this experiment, you should ensure that you have answered all previous questions from the last section and you've pressed "Enter" on the final experiment from evaluation 2.

Run the final evaluation:

Hit "Play"


Activities
Firefox Web Browser
Mon 18 Mar 19:09
Participant 0 - Colaborator
https://colab.research.google.com/drive/12nliNP3hW44exlq3f169aGPgrgFnwpuj#scrollTo=WiwMkpP6_z-D&uniqifier=2
Participant 0
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
RAM
Disk
EDITING

```
[1] --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: ‘evaluations.tar.gz’

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - ‘evaluations.tar.gz’ saved [2037760/2037760]

    Ready participant 0!

[2] Evaluation 1

[ ] Evaluation 2

    Evaluation 3
```

Again, repeat the procedure from previous sections.

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

22. **3 - Experiment 1** *

Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

23. **3 - Experiment 2** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

24. **3 - Experiment 3** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

25. **3 - Experiment 4** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

26. **3 - Experiment 5** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

27. **3 - Experiment 6** *
Mark only one oval.

- ○ 1
- ○ 2
- ○ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

28. **3 - Experiment 7** *
Mark only one oval.

- ○ 1
- ○ 2
- ○ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

29. **3 - Experiment 8** *
Mark only one oval.

- ○ 1
- ○ 2
- ○ 3

Thank you!

The main aim of the experiment was to evaluate whether a deep learning method I'd come up with could allow us to conditionally generate music. I hoped the technique would be able to capture some of the latent features in music that make up, e.g. a compositional style.

In the first experiment, you were listening to three samples of music. In each set of samples, one was from a human, one was from a baseline model (something I can use to compare my technique to), and one was from my model. I was hoping to see that you picked the baseline model more frequently than my model.

In the second experiment, you listened to three samples of human music. Two of the three samples in each set were from a single composer, and one was from a different composer. I was trying to see if people could actually discern different composer styles.

In the final experiment, you listened to three samples of music. In each set of samples, one was from a human, one was from a baseline model and one was from my model. Each of the samples was supposed to all sound like the same compositional style. I was hoping that my model was able to better capture compositional style, and so the baseline would be selected as the odd one out since it didn't sound like the human or my model.

Please take note of my email and that of my supervisor's. If you have any questions or comments feel free to reach out, or if you'd like to see the source code used in these experiments. Thanks for participating!

salman.mohammadi@outlook.com

Supervisor:
Bjorn Sand Jensen
bjorn.jensen@glasgow.ac.uk

Exploring Artificial Intelligence and Music

CS4, 2018/2019
Salman Mohammadi
salman.mohammadi@outlook.com

The aim of this experiment is to evaluate a novel deep learning method for conditionally generating music, i.e. trying to get a computer to compose music which has a certain style or attribute. It's difficult to evaluate how good the music is unless we ask humans to listen to it, which is why we require human involvement. In the experiment, you will be asked to answer a survey which outlines your musical background, and then listen to samples of music and answer some mulitple-choice questions. All data collected will be anonymous, and you are welcome to withdraw from the test at any point. Please feel free to ask any questions now before the start of the experiment using the contact details provided.

This is not a test of your ability! We're evaluating the model, not how well you are able to answer the questions.

The evaluation should take no more than 10 minutes.

If you consent to taking part in this experiment, enter your provided participant number below, press next and follow the instructions provided.

* Required

1. **Participant number** *

____

Musical Background

This section will aim to assess your prior musical experience and training. If you're unsure about any of the questions, be conservative with your estimates. As stated previously, this is not a test of your ability.

2. **How many years of formal musical training do you have? (0 if none)** *

____

3. **Can you read sheet music?** *

Mark only one oval.

- Yes
- No

4. **Is English your first langauge?** *
   Mark only one oval.

   ⬭ Yes
   ⬭ No
   ⬭ I don't know

5. **Did you grow up listening to mostly English music?** *
   Mark only one oval.

   ⬭ Yes
   ⬭ No
   ⬭ I don't know

6. **Do you regularly listen to and enjoy classical music?** *
   Mark only one oval.

   ⬭ Yes
   ⬭ No

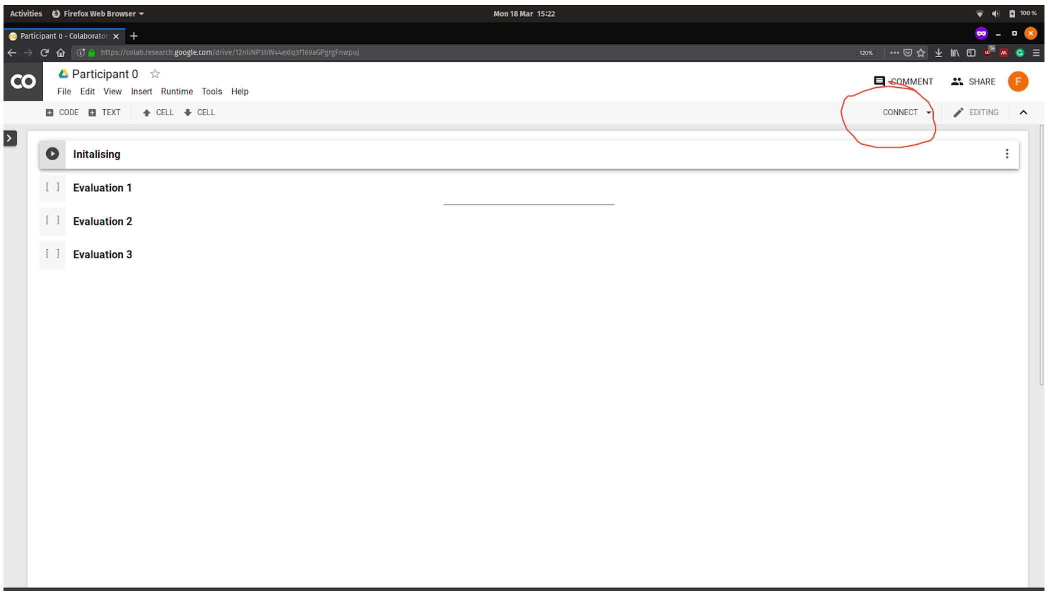
7. **Tick all of the following composers which you would be able to distinctly recognise musical pieces by.**
   Check all that apply.

   ☐ Chopin
   ☐ Bach
   ☐ Beethoven
   ☐ Debussy

Experiment Setup

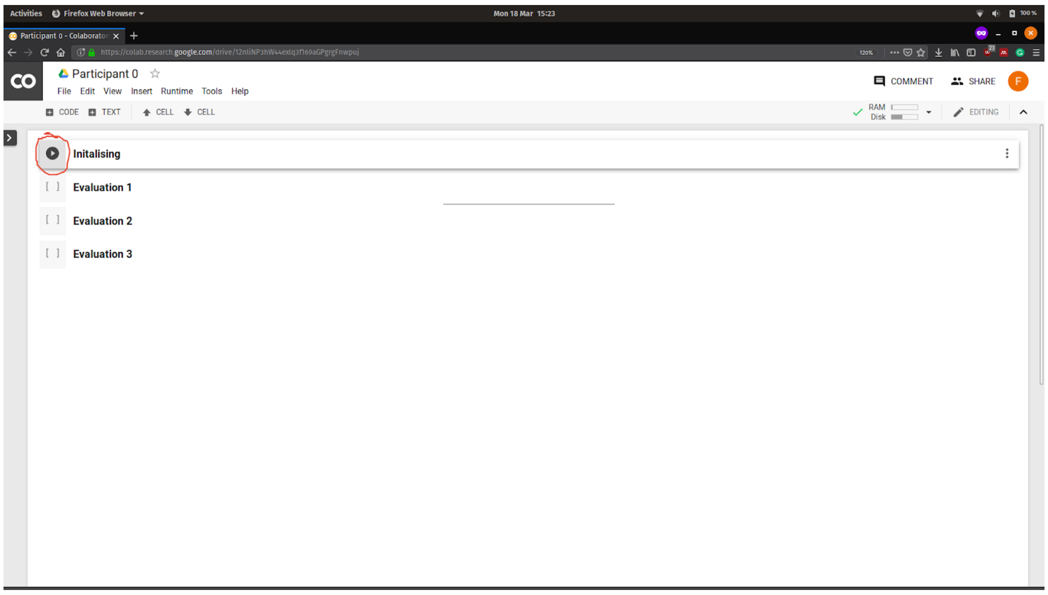
At this point you should open the link I sent along with this form. You'll need a google account to be able to run the experiments, but rest assured all data is anonymous and you only need to log into be able to use google's free online notebook platform.

Click "Connect" in the top right



Once it's done (it may take a while), you should see the image below. Follow the next step.

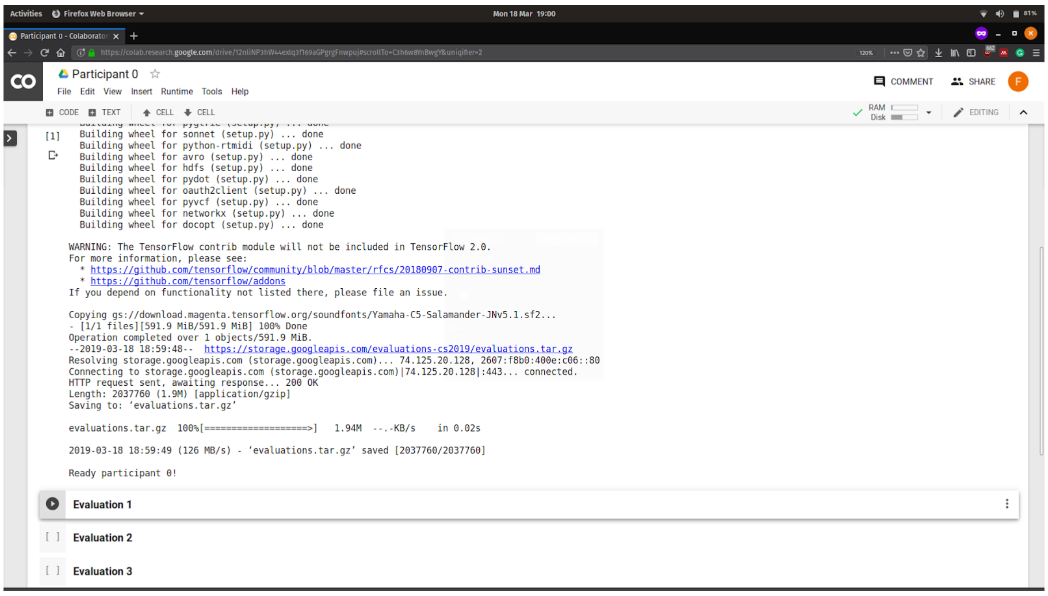
Hit the "Play" arrow shaped button highlighted.



You might get a warning about the notebook not being authored by Google. Hit "Run Anyway" to proceed. If another pop-up appears saying "Reset all runtimes", hit Yes.

I'm happy to share full source code after the experiment.

Once it's done running (it may take a minute or two), you should see the following screen.



```
  Building wheel for sonnet (setup.py) ... done
  Building wheel for python-rtmidi (setup.py) ... done
  Building wheel for avro (setup.py) ... done
  Building wheel for hdfs (setup.py) ... done
  Building wheel for pydot (setup.py) ... done
  Building wheel for oauth2client (setup.py) ... done
  Building wheel for pyvcf (setup.py) ... done
  Building wheel for networkx (setup.py) ... done
  Building wheel for docopt (setup.py) ... done

WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
If you depend on functionality not listed there, please file an issue.

Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
- [1/1 files][591.9 MiB/591.9 MiB] 100% Done
Operation completed over 1 objects/591.9 MiB.
--2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2037760 (1.9M) [application/gzip]
Saving to: 'evaluations.tar.gz'

evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

2019-03-18 18:59:49 (126 MB/s) - 'evaluations.tar.gz' saved [2037760/2037760]

Ready participant 0!
```

You can now proceed to the first evaluation, press next below to begin.

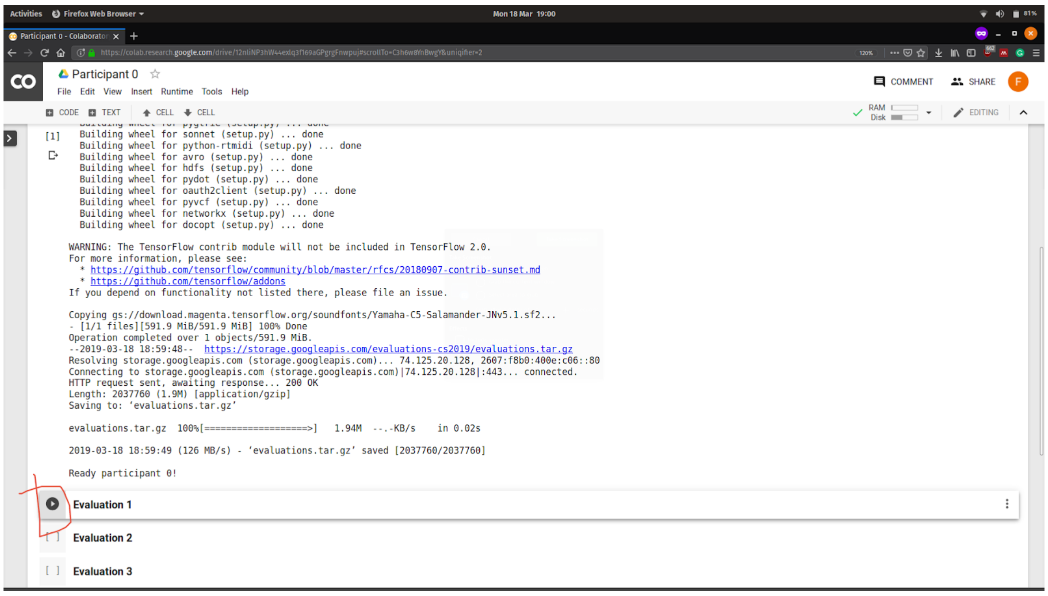
Evaluation 1

You will be given 8 experiments with three samples of music each, each sample 10 seconds long. In each experiment, you will be given an "anchor" sample, which is a human. You must select one sample from the remaining two that sounds LEAST like it was made by a human.

The instructions for this evaluation are as follows:

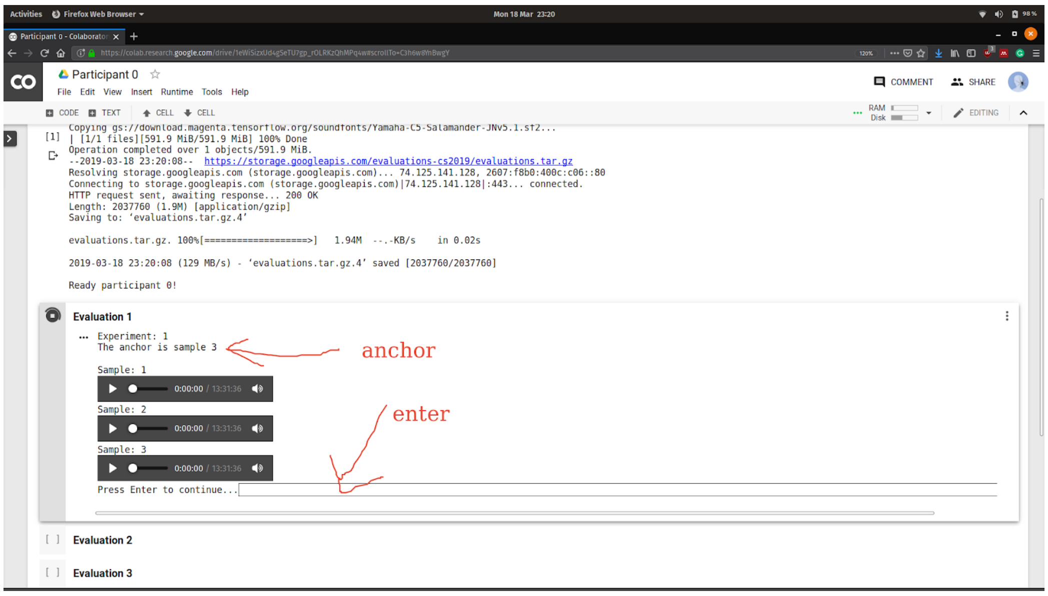
To run the evaluation

Hit the "Play" arrow shaped button highlighted.



```
  Building wheel for sonnet (setup.py) ... done
  Building wheel for python-rtmidi (setup.py) ... done
  Building wheel for avro (setup.py) ... done
  Building wheel for hdfs (setup.py) ... done
  Building wheel for pydot (setup.py) ... done
  Building wheel for oauth2client (setup.py) ... done
  Building wheel for pyvcf (setup.py) ... done
  Building wheel for networkx (setup.py) ... done
  Building wheel for docopt (setup.py) ... done

WARNING: The TensorFlow contrib module will not be included in TensorFlow 2.0.
For more information, please see:
  * https://github.com/tensorflow/community/blob/master/rfcs/20180907-contrib-sunset.md
  * https://github.com/tensorflow/addons
If you depend on functionality not listed there, please file an issue.

Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
- [1/1 files][591.9 MiB/591.9 MiB] 100% Done
Operation completed over 1 objects/591.9 MiB.
--2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2037760 (1.9M) [application/gzip]
Saving to: 'evaluations.tar.gz'

evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

2019-03-18 18:59:49 (126 MB/s) - 'evaluations.tar.gz' saved [2037760/2037760]

Ready participant 0!
```

Press the play button on each of the samples. You can listen to each one as many times as you like.


Activities
Firefox Web Browser
Mon 18 Mar 23:20
Participant 0 - Colaborator
https://colab.research.google.com/drive/1eWiSizxUd4gSeTU7gp_rOLRKzQhMPq4w#scrollTo=C3h6w8YnBwgY
Participant 0
File Edit View Insert Runtime Tools Help
CODE TEXT CELL CELL
COMMENT SHARE
RAM Disk
EDITING

```
Copying gs://download.magenta.tensorflow.org/soundfonts/Yamaha-C5-Salamander-JNv5.1.sf2...
| [1/1 files][591.9 MiB/591.9 MiB] 100% Done
Operation completed over 1 objects/591.9 MiB.
--2019-03-18 23:20:08--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.141.128, 2607:f8b0:400c:c06::80
Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.141.128|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 2037760 (1.9M) [application/gzip]
Saving to: ‘evaluations.tar.gz.4’

evaluations.tar.gz. 100%[===================>]   1.94M  --.-KB/s    in 0.02s

2019-03-18 23:20:08 (129 MB/s) - ‘evaluations.tar.gz.4’ saved [2037760/2037760]

Ready participant 0!
```

Evaluation 1

```
Experiment: 1
The anchor is sample 3

Sample: 1
0:00:00 / 13:31:36
Sample: 2
0:00:00 / 13:31:36
Sample: 3
0:00:00 / 13:31:36
Press Enter to continue...
```

Evaluation 2

Evaluation 3

In the example above, the user must select one from the remaining samples (e.g. 1 or 2) that sounds least similar to sample 3. Follow the instructions and use your own link to complete the evaluation.

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

8. **1 - Experiment 1** *

Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Once you're done, fill out the question for the corresponding experiment, and click in the box shown and press Enter.

The next set of samples for the next experiment will now be available to listen to. After answering each experiment, click in the box and press enter to proceed to the next question.

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

9. **1 - Experiment 2** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

10. **1 - Experiment 3** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

11. **1 - Experiment 4** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

12. **1 - Experiment 5** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

13. **1 - Experiment 6** *

Mark only only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

14. **1 - Experiment 7** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by a human.

15. **1 - Experiment 8** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

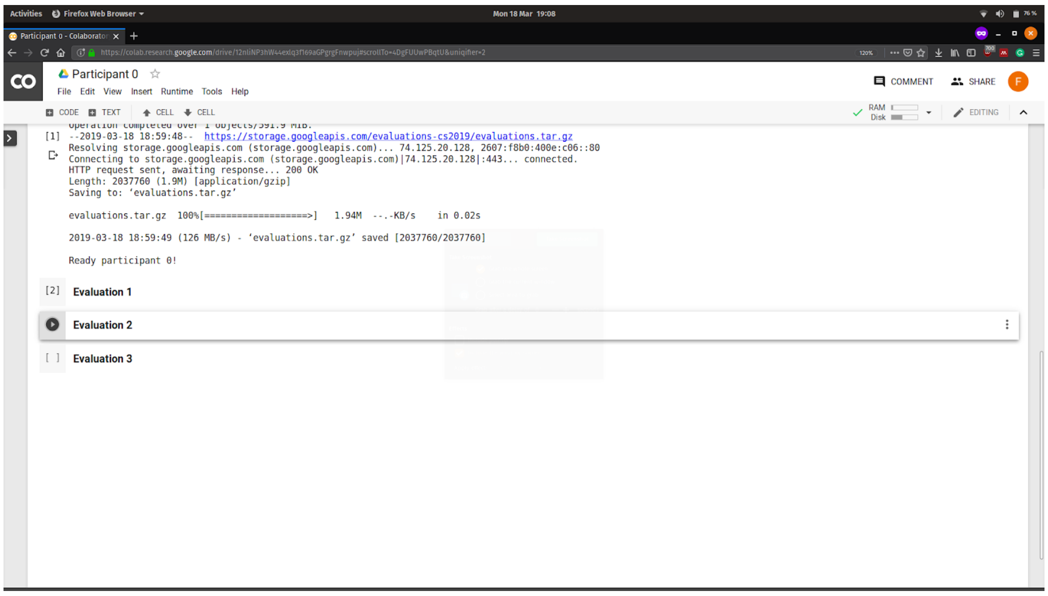
Evaluation 2

You will be given 6 experiments with three samples of music each, each sample 10 seconds long and created by a human. In each experiment, you will be given an "anchor" sample. You must select one sample from the remaining two that sounds LEAST like it came from the same composer as the anchor.

To run the evaluation:

To begin this experiment, you should ensure that you have answered all previous questions from the last section and you've pressed "Enter" on the final experiment from evaluation 1.

Hit "Play" to begin this evaluation.


Activities
Firefox Web Browser
Mon 18 Mar 19:08
Participant 0 - Colaborator
https://colab.research.google.com/drive/12nliNP3hW44exlq3f169aGPgrgFnwpuj#scrollTo=4DgFUUwPBqtU&uniqifier=2
Participant 0
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
RAM
Disk
EDITING

```
[1] --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: ‘evaluations.tar.gz’

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - ‘evaluations.tar.gz’ saved [2037760/2037760]

    Ready participant 0!
```

[2] **Evaluation 1**

Evaluation 2

[ ] **Evaluation 3**

The instructions are exactly the same as last time: listen to the samples, answer the corresponding question, and hit Enter to proceed.

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

16. **2 - Experiment 1** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

17. **2 - Experiment 2** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

18. **2 - Experiment 3** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

19. **2 - Experiment 4** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

20. **2 - Experiment 5** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Select the sample which is not the anchor, that sounds LEAST like it was made by the same composer as the anchor.

21. **2 - Experiment 6** *

Mark only one oval.

- ( ) 1
- ( ) 2
- ( ) 3

Evaluation 3

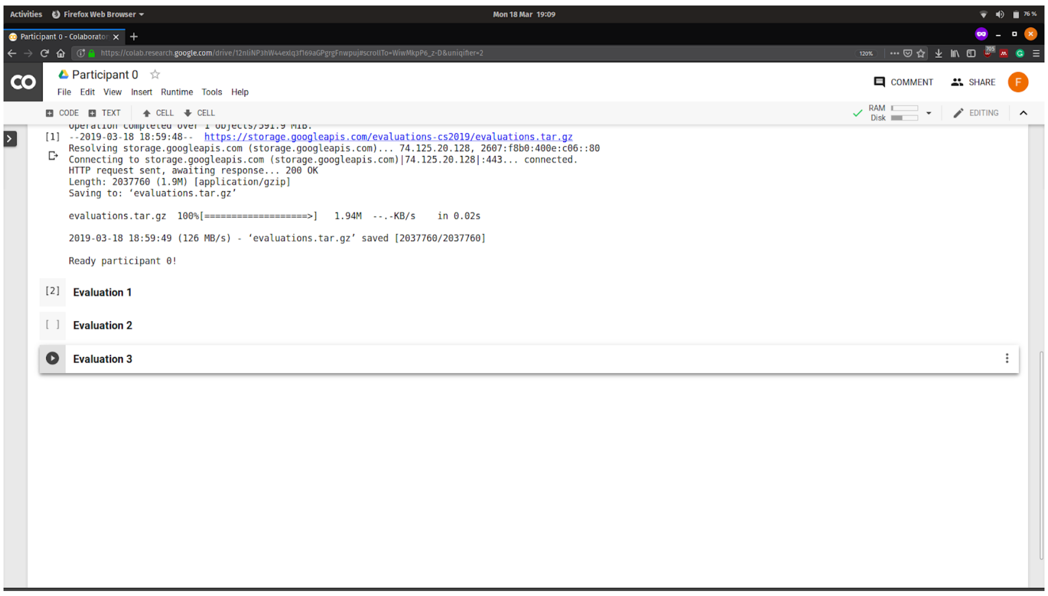
You will be given 8 experiments with three samples of music each, each sample 10 seconds long. In each experiment, you will be given an "anchor" sample. You must select one sample from the remaining two that sounds LEAST like the anchor sample.

To run the evaluation:

To begin this experiment, you should ensure that you have answered all previous questions from the last section and you've pressed "Enter" on the final experiment from evaluation 2.

Run the final evaluation:

Hit "Play"


Activities
Firefox Web Browser
Mon 18 Mar 19:09
Participant 0 - Colaborator
https://colab.research.google.com/drive/12nliNP3hW44exlq3f169aGPgrgFnwpuj#scrollTo=WiwMkpP6_z-D&uniqifier=2
Participant 0
File Edit View Insert Runtime Tools Help
COMMENT
SHARE
CODE
TEXT
CELL
CELL
RAM
Disk
EDITING

```
[1] --2019-03-18 18:59:48--  https://storage.googleapis.com/evaluations-cs2019/evaluations.tar.gz
    Resolving storage.googleapis.com (storage.googleapis.com)... 74.125.20.128, 2607:f8b0:400e:c06::80
    Connecting to storage.googleapis.com (storage.googleapis.com)|74.125.20.128|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 2037760 (1.9M) [application/gzip]
    Saving to: 'evaluations.tar.gz'

    evaluations.tar.gz  100%[===================>]   1.94M  --.-KB/s    in 0.02s

    2019-03-18 18:59:49 (126 MB/s) - 'evaluations.tar.gz' saved [2037760/2037760]

    Ready participant 0!

[2] Evaluation 1

[ ] Evaluation 2

    Evaluation 3
```

Again, repeat the procedure from previous sections.

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

22. **3 - Experiment 1** *

Mark only one oval.

- ⬭ 1
- ⬭ 2
- ⬭ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

23. **3 - Experiment 2** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

24. **3 - Experiment 3** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

25. **3 - Experiment 4** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

26. **3 - Experiment 5** *

Mark only one oval.

- 1
- 2
- 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

27. **3 - Experiment 6** *
Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

28. **3 - Experiment 7** *
Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Select the sample which is not the anchor, that sounds LEAST similar to the anchor.

29. **3 - Experiment 8** *
Mark only one oval.

- ◯ 1
- ◯ 2
- ◯ 3

Thank you!

The main aim of the experiment was to evaluate whether a deep learning method I'd come up with could allow us to conditionally generate music. I hoped the technique would be able to capture some of the latent features in music that make up, e.g. a compositional style.

In the first experiment, you were listening to three samples of music. In each set of samples, one was from a human, one was from a baseline model (something I can use to compare my technique to), and one was from my model. I was hoping to see that you picked the baseline model more frequently than my model.

In the second experiment, you listened to three samples of human music. Two of the three samples in each set were from a single composer, and one was from a different composer. I was trying to see if people could actually discern different composer styles.

In the final experiment, you listened to three samples of music. In each set of samples, one was from a human, one was from a baseline model and one was from my model. Each of the samples was supposed to all sound like the same compositional style. I was hoping that my model was able to better capture compositional style, and so the baseline would be selected as the odd one out since it didn't sound like the human or my model.

Please take note of my email and that of my supervisor's. If you have any questions or comments feel free to reach out, or if you'd like to see the source code used in these experiments. Thanks for participating!

salman.mohammadi@outlook.com

Supervisor:
Bjorn Sand Jensen
bjorn.jensen@glasgow.ac.uk

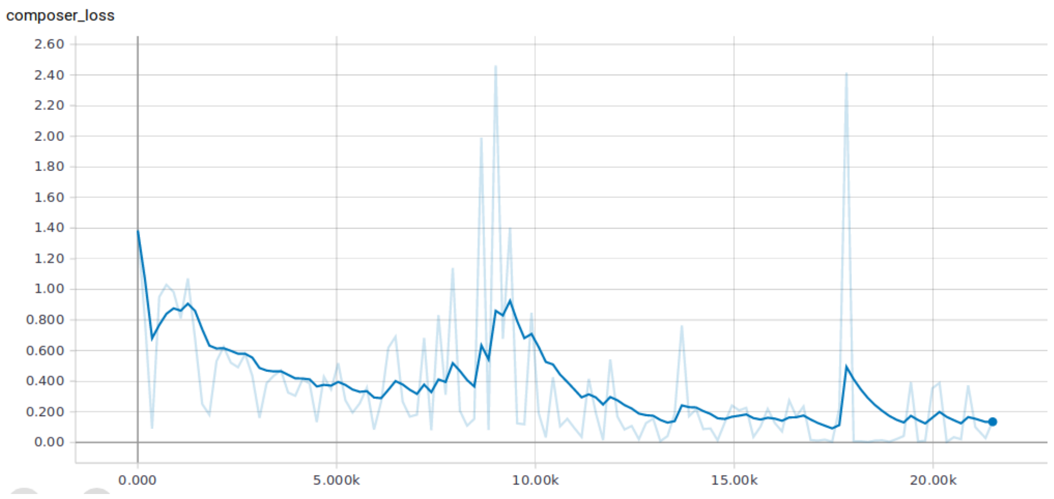
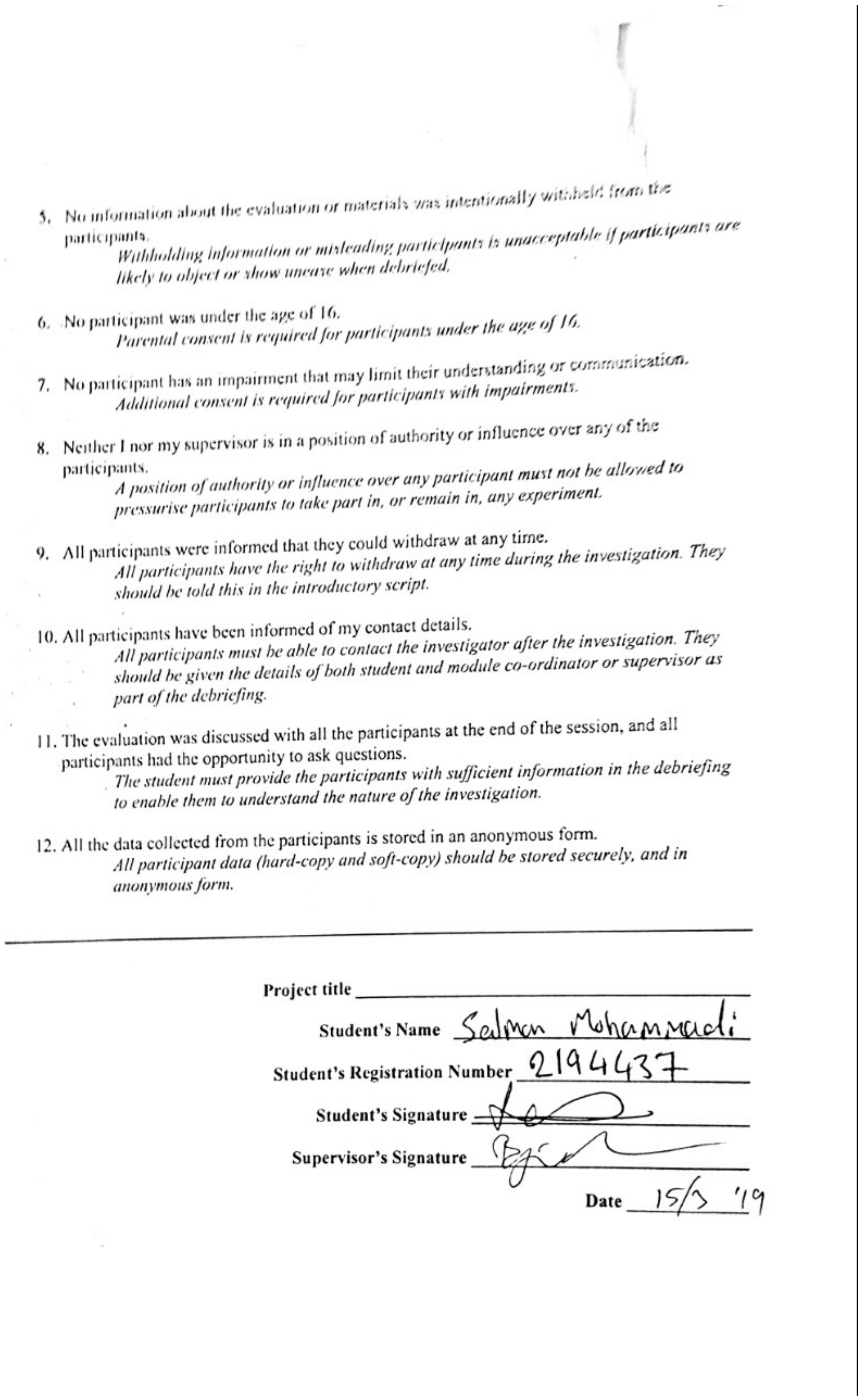


Figure A.3: *Accuracy for the composer classifier cross-entropy for CondRNN over 22,000 steps.*

Online samples to listen to the models: A link to the colaboratory notebook: https://colab.research.google.com/drive/1ch-M9bsu9SI-6qfCz9yHlok0JYV43Okl

A.3 Samples

https://storage.googleapis.com/evaluations-cs2019/samples.zip



5. No information about the evaluation or materials was intentionally withheld from the participants.
   Withholding information or misleading participants is unacceptable if participants are likely to object or show unease when debriefed.

6. No participant was under the age of 16.
   Parental consent is required for participants under the age of 16.

7. No participant has an impairment that may limit their understanding or communication.
   Additional consent is required for participants with impairments.

8. Neither I nor my supervisor is in a position of authority or influence over any of the participants.
   A position of authority or influence over any participant must not be allowed to pressurise participants to take part in, or remain in, any experiment.

9. All participants were informed that they could withdraw at any time.
   All participants have the right to withdraw at any time during the investigation. They should be told this in the introductory script.

10. All participants have been informed of my contact details.
    All participants must be able to contact the investigator after the investigation. They should be given the details of both student and module co-ordinator or supervisor as part of the debriefing.

11. The evaluation was discussed with all the participants at the end of the session, and all participants had the opportunity to ask questions.
    The student must provide the participants with sufficient information in the debriefing to enable them to understand the nature of the investigation.

12. All the data collected from the participants is stored in an anonymous form.
    All participant data (hard-copy and soft-copy) should be stored securely, and in anonymous form.

Project title __________

Student's Name Salman Mohammadi

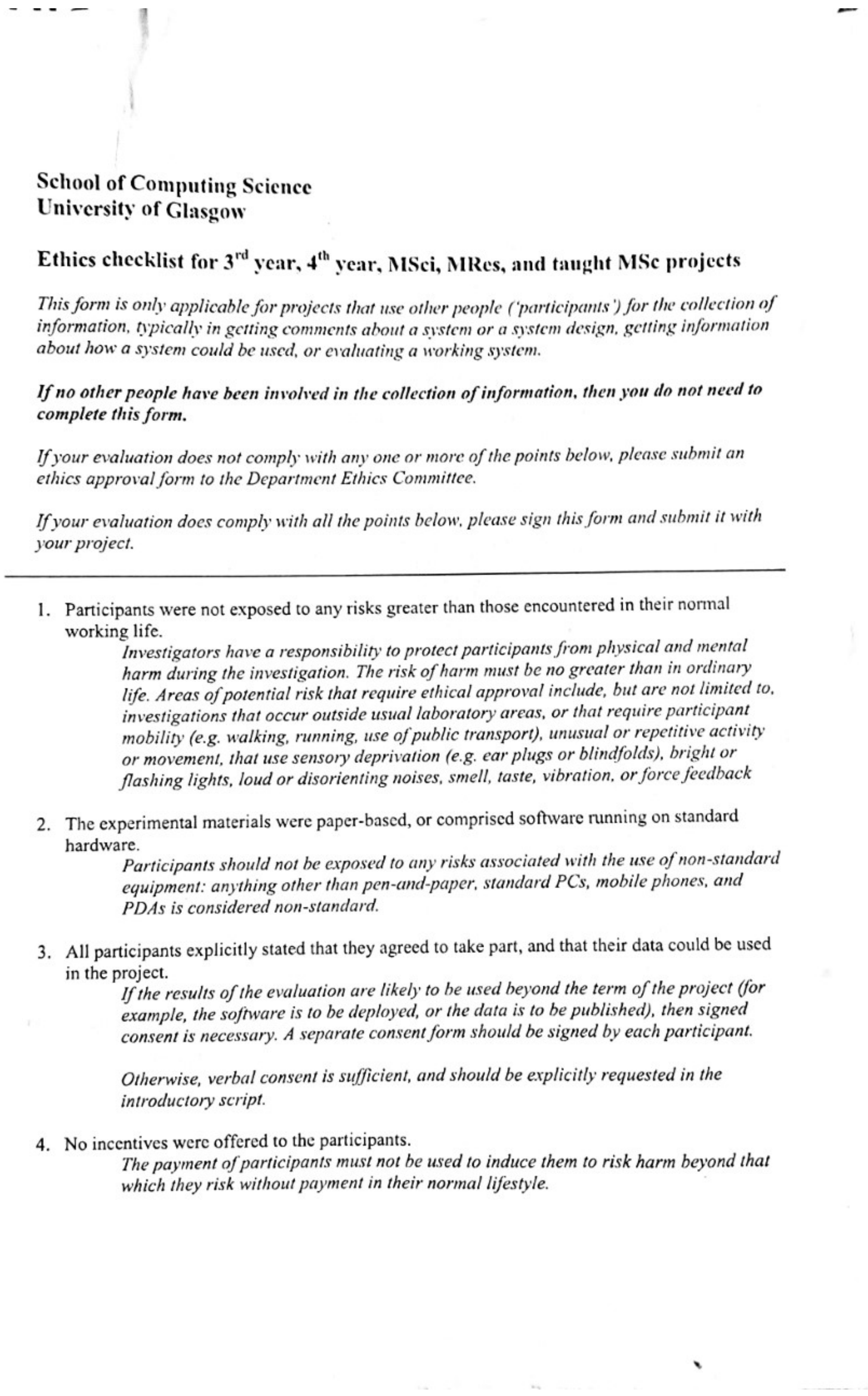
Student's Registration Number 2194437

Student's Signature [illegible]

Supervisor's Signature [illegible]

Date 15/3 '19

Figure A.4: Ethics Approval



School of Computing Science
University of Glasgow

Ethics checklist for 3rd year, 4th year, MSci, MRes, and taught MSc projects

This form is only applicable for projects that use other people ('participants') for the collection of information, typically in getting comments about a system or a system design, getting information about how a system could be used, or evaluating a working system.

If no other people have been involved in the collection of information, then you do not need to complete this form.

If your evaluation does not comply with any one or more of the points below, please submit an ethics approval form to the Department Ethics Committee.

If your evaluation does comply with all the points below, please sign this form and submit it with your project.

1. Participants were not exposed to any risks greater than those encountered in their normal working life.
   Investigators have a responsibility to protect participants from physical and mental harm during the investigation. The risk of harm must be no greater than in ordinary life. Areas of potential risk that require ethical approval include, but are not limited to, investigations that occur outside usual laboratory areas, or that require participant mobility (e.g. walking, running, use of public transport), unusual or repetitive activity or movement, that use sensory deprivation (e.g. ear plugs or blindfolds), bright or flashing lights, loud or disorienting noises, smell, taste, vibration, or force feedback

2. The experimental materials were paper-based, or comprised software running on standard hardware.
   Participants should not be exposed to any risks associated with the use of non-standard equipment: anything other than pen-and-paper, standard PCs, mobile phones, and PDAs is considered non-standard.

3. All participants explicitly stated that they agreed to take part, and that their data could be used in the project.
   If the results of the evaluation are likely to be used beyond the term of the project (for example, the software is to be deployed, or the data is to be published), then signed consent is necessary. A separate consent form should be signed by each participant.

   Otherwise, verbal consent is sufficient, and should be explicitly requested in the introductory script.

4. No incentives were offered to the participants.
   The payment of participants must not be used to induce them to risk harm beyond that which they risk without payment in their normal lifestyle.

Figure A.5: *Ethics Approval*

8 Bibliography

H. Anton and C. Rorres. *Elementary linear algebra*. 1994. URL https://trove.nla.gov.au/work/5455951?q{&}versionId=13296821.

D. Bahdanau, K. Cho, and Y. Bengio. Neural Machine Translation by Jointly Learning to Align and Translate. sep 2014. URL http://arxiv.org/abs/1409.0473.

S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Jozefowicz, and S. Bengio. Generating Sentences from a Continuous Space. nov 2015. ISSN 10450823. doi: 10.18653/v1/K16-1002. URL http://arxiv.org/abs/1511.06349.

C. M. Brown and P. Hagoort. The Neurocognition of Language, 2000. URL https://philpapers.org/rec/MBRTNO.

R. Caruana. Multitask Learning. *Mach. Learn.*, 28(1):41–75, 1997. ISSN 08856125. doi: 10.1023/A:1007379606734. URL http://link.springer.com/10.1023/A:1007379606734.

C. Darwin. *The Descent of Man and Selection in Relation to Sex*. 1871. URL http://www.gutenberg.org/files/2300/2300-h/2300-h.htm{#}link2HCH0003.

S. Dieleman, A. van den Oord, and K. Simonyan. The challenge of realistic music generation: modelling raw audio at scale. jun 2018. URL http://arxiv.org/abs/1806.10474.

C. Donahue, J. McAuley, and M. Puckette. Adversarial Audio Synthesis. feb 2018. ISSN 14341948. doi: 1802.04208. URL http://arxiv.org/abs/1802.04208.

C. Drake and D. Bertrand. The quest for universals in temporal processing in music. *Ann. N. Y. Acad. Sci.*, 930:17–27, jun 2001. ISSN 0077-8923. URL http://www.ncbi.nlm.nih.gov/pubmed/11458828.

N. R. Draper. The Cambridge Dictionary of Statistics, Fourth Edition by B. S. Everitt, A. Skrondal. *Int. Stat. Rev.*, 79(2):273–274, aug 2011. ISSN 03067734. doi: 10.1111/j.1751-5823.2011.00149_2.x. URL http://doi.wiley.com/10.1111/j.1751-5823.2011.00149{_}2.x.

D. Eck, J. S. I. D. M. D. S. S. Intelligenza, and undefined 2002. A first look at music composition using lstm recurrent neural networks. *people.idsia.ch*. URL http://people.idsia.ch/{~}juergen/blues/IDSIA-07-02.pdf.

J. Engel, C. Resnick, A. Roberts, S. Dieleman, D. Eck, K. Simonyan, and M. Norouzi. Neural Audio Synthesis of Musical Notes with WaveNet Autoencoders. apr 2017. URL http://arxiv.org/abs/1704.01279.

F. A. Gers, J. Schmidhuber, and F. Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Comput.*, 12(10):2451–2471, oct 2000. ISSN 0899-7667. doi: 10.1162/089976600300015015. URL http://www.mitpressjournals.org/doi/10.1162/089976600300015015.

I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning*. 2016. URL https://books.google.co.uk/books?hl=en{&}lr={&}id=omivDQAAQBAJ{&}oi=fnd{&}pg=PR5{&}dq=deep+learning+ian+goodfellow{&}ots=MMR1ctsGRU{&}sig=eA3mBvP65XqPKCaAugRRO5h7Cic.

Google. Colaboratory. URL https://colab.research.google.com/notebooks/welcome.ipynb{#}recent=true.

Google. Google Forms, 2019. URL https://www.google.com/forms/about/.

GoogleCloud. Google Cloud Platform. URL https://console.cloud.google.com/home/dashboard?project=midi-maestro.

K. Gregor, I. Danihelka, A. Mnih, C. Blundell, and D. Wierstra. Deep AutoRegressive Networks. oct 2013. URL http://arxiv.org/abs/1310.8499.

C. Gulcehre, O. Firat, K. Xu, K. Cho, L. Barrault, H.-C. Lin, F. Bougares, H. Schwenk, and Y. Bengio. On Using Monolingual Corpora in Neural Machine Translation. mar 2015. URL http://arxiv.org/abs/1503.03535.

C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck. Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset. oct 2018. doi: arXiv:1810.12247v2. URL http://arxiv.org/abs/1810.12247.

G. Hinton. Neural Networks for Machine Learning: Slides. URL http://www.cs.toronto.edu/{~}tijmen/csc321/slides/lecture{_}slides{_}lec6.pdf.

S. Hochreiter, S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient Flow in Recurrent Nets: the Difficulty of Learning Long-Term Dependencies. 2001. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.24.7321.

C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. Music Transformer. sep 2018. URL http://arxiv.org/abs/1809.04281.

D. Huron. Is Music an Evolutionary Adaptation? In *Cogn. Neurosci. Music*, volume 930, pages 43–61. John Wiley & Sons, Ltd (10.1111), jun 2012. ISBN 9780191689314. doi: 10.1093/acprof:oso/9780198525202.003.0005. URL http://doi.wiley.com/10.1111/j.1749-6632.2001.tb05724.x.

D. B. Huron. *Sweet anticipation : music and the psychology of expectation*. MIT Press, 2006. ISBN 9780262083454.

IFPI. Global Music Report âĂŤ IFPI âĂŤ Representing the recording industry worldwide, 2018. URL https://www.ifpi.org/downloads/GMR2018.pdf.

Jupyter. Project Jupyter | Home. URL https://jupyter.org/.

S. Karayev, E. Shelhamer, S. Guadarrama, R. Girshick, Y. Jia, T. Darrell, J. Long, and J. Donahue. Caffe. pages 675–678, may 2014. ISSN 9781450330633. doi: 10.1145/2647868.2654889. URL http://arxiv.org/abs/1506.00019.

Laurens van der Maaten and G. Hinton. Visualizing Data using t-SNE Laurens. *J. Mach. Learn. Res.*, 9(Nov):2579–2605, 2008. ISSN 02624079. doi: 10.1007/s10479-011-0841-3. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

H. T. Lawless and H. Heymann. Discrimination Testing. In *Sens. Eval. Food*, pages 116–139. Springer US, Boston, MA, 1999. doi: 10.1007/978-1-4615-7843-7_4. URL http://link.springer.com/10.1007/978-1-4615-7843-7{_}4.

J. Lewis. Creation by refinement: a creativity paradigm for gradient descent learning networks - IEEE Conference Publication. 1988. URL https://ieeexplore.ieee.org/document/23933.

F. Liang, M. Gotham, M. Johnson, and J. Shotton. Automatic Stylistic Composition of Bach Chorales with Deep LSTM. *Proc. 18th Int. Soc. Music Inf. Retr. Conf.*, pages 449–456, 2017. URL https://ismir2017.smcnus.org/wp-content/uploads/2017/10/156{_}Paper.pdf.

Magenta. Magenta: Music and Art Generation with Machine Intelligence. URL https://github.com/tensorflow/magenta.

J. McDermott and M. Hauser. The Origins of Music, Innateness, Uniqueness, and Evolution. 23 (1):29–59, 2005.

S. Mithen. *The Singing Neanderthals: The Origins of Music, Language, Mind, and Body (Harvard Univ Press, Cambridge, MA).* 2006. ISBN 9780674025592. URL http://www.hup.harvard.edu/catalog.php?isbn=9780674025592.

J. Montagu. How Music and Instruments Began: A Brief Overview of the Origin and Entire Development of Music, from Its Earliest Stages. *Front. Sociol.*, 2:8, jun 2017. ISSN 2297-7775. doi: 10.3389/fsoc.2017.00008. URL http://journal.frontiersin.org/article/10.3389/fsoc.2017.00008/full.

N. Mor, L. Wolf, A. Polyak, and Y. Taigman. A Universal Music Translation Network. Technical report.

V. Mottier. âĂĲTalking about music is like dancing about architectureâĂİ: Artspeak and pop music. *Lang. Commun.*, 29(2):127–132, apr 2009. ISSN 0271-5309. doi: 10.1016/J.LANGCOM.2009.01.003. URL https://www.sciencedirect.com/science/article/pii/S027153090900010X.

M. C. Mozer. Neural Network Music Composition by Prediction: Exploring the Benefits of Psychoacoustic Constraints and Multi-scale Processing. *Conn. Sci.*, 6(2-3):247–280, jan 1994. ISSN 0954-0091. doi: 10.1080/09540099408915726. URL https://www.tandfonline.com/doi/full/10.1080/09540099408915726.

M. Nielsen. *Neural networks and deep learning*. 2015. URL http://static.latexstudio.net/article/2018/0912/neuralnetworksanddeeplearning.pdf.

C. Olah. Understanding Convolutions - colah's blog, 2014. URL http://colah.github.io/posts/2014-07-Understanding-Convolutions/.

C. Olah. Understanding LSTM Networks, 2015. URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

J. P. Olive, C. Christianson, and J. McCary. *Handbook of natural language processing and machine translation : DARPA global autonomous language exploitation.* Springer, 2011. ISBN 9781441977120.

S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This Time with Feeling: Learning Expressive Musical Performance. pages 1–24, 2018. URL http://arxiv.org/abs/1808.03715.

R. Pascanu, T. Mikolov, and Y. Bengio. On the difficulty of training Recurrent Neural Networks. nov 2012. URL http://arxiv.org/abs/1211.5063.

A. D. Patel. Language, music, syntax and the brain. *Nat. Neurosci.*, 6(7):674–681, jul 2003. ISSN 10976256. doi: 10.1038/nn1082. URL http://www.nature.com/articles/nn1082.

C. Raffel. Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching. 2016. doi: 10.7916/D8N58MHV. URL `https://academiccommons.columbia.edu/doi/10.7916/D8N58MHV`.

W. Rawat and Z. Wang. Deep Convolutional Neural Networks for Image Classification: A Comprehensive Review. *Neural Comput.*, 29(9):2352–2449, sep 2017. ISSN 0899-7667. doi: 10.1162/neco_a_00990. URL `http://www.mitpressjournals.org/doi/abs/10.1162/neco{_}a{_}00990`.

A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music, jul 2018. ISSN 1938-7228. URL `http://proceedings.mlr.press/v80/roberts18a.html`.

S. Sargur. RNNs: Teacher Forcing. URL `https://cedar.buffalo.edu/{~}srihari/CSE676/10.2.1TeacherForcing.pdf`.

A. Schindler, R. Mayer, and A. Rauber. Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset. *Ismir*, (Ismir):469–474, 2012. URL `http://ismir2012.ismir.net/event/papers/469-ismir-2012.pdf`.

S. S. Sealy. J.S. Bach: A Study in Musical Symmetry. URL `http://www.math.brown.edu/{~}banchoff/Yale/project04/bach.html`.

C. Sinkinson. Triangle Test. *Discrim. Test. Sens. Sci.*, pages 153–170, jan 2017. doi: 10.1016/B978-0-08-101009-9.00007-1. URL `https://www.sciencedirect.com/science/article/pii/B9780081010099000071`.

H. Stone and J. L. Sidel. *Sensory evaluation practices*. Elsevier Academic Press, 2004. ISBN 9780126726909.

B. Sturm. Even more Endless Music Sessions | High Noon GMT, 2017. URL `https://highnoongmt.wordpress.com/2017/06/14/even-more-endless-music-sessions/`.

A. Swift. A brief introduction to MIDI, 1997. URL `http://www.doc.ic.ac.uk/{~}nd/surprise{_}97/journal/vol1/aps2/`.

L. Theis, A. van den Oord, and M. Bethge. A note on the evaluation of generative models. nov 2015. URL `http://arxiv.org/abs/1511.01844`.

Todd. A sequential network design for musical applications. 1988.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel Recurrent Neural Networks. jan 2016. ISSN 0277786X. doi: 10.1117/12.860537. URL `http://arxiv.org/abs/1601.06759`.

A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention Is All You Need. Technical report.

S. Venugopalan, L. A. Hendricks, R. Mooney, and K. Saenko. Improving LSTM-based Video Description with Linguistic Knowledge Mined from Text. apr 2016. URL `http://arxiv.org/abs/1604.01729`.

P. Vuust and C. D. Frith. Anticipation is the key to understanding music and the effects of music on emotion. *Behav. Brain Sci.*, 31(05):599–600, oct 2008. ISSN 0140-525X. doi: 10.1017/S0140525X08005542. URL `http://www.journals.cambridge.org/abstract{_}S0140525X08005542`.

C. Walshaw. ABC. URL `http://abcnotation.com/`.