Dissertation Type: research

DEPARTMENT OF COMPUTER SCIENCE

Performing Algorithmic Co-composition Using Machine Learning

Stephen Livermore-Tozer

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of Master of Engineering in the Faculty of Engineering.

Saturday 23rd April, 2016

Declaration

This dissertation is submitted to the University of Bristol in accordance with the requirements of the degree of MEng in the Faculty of Engineering. It has not been submitted for any other degree or diploma of any examining body. Except where specifically acknowledged, it is all the work of the Author.

Stephen Livermore-Tozer, Saturday 23rd April, 2016

Contents

List of Listings

Executive Summary

For my research, I have investigated the hypothesis that existing methods of performing melodic composition using machine learning can be adapted to successfully perform co-composition with a human composer. In this context, success may be measured by subjective impression received from experts in composition. The exact set of tasks involved in the co-composition process is not fixed; part of this research involved finding the tasks that are both a) useful to the composer, and b) within the capabilities of current AI.

In the pursuit of this objective, I have completed the following tasks:

- I surveyed a number of expert composers to identify the tasks within the domain of composition whose partial automation would provide the greatest benefit to the composer.
- I performed research over existing machine learning methods for algorithmic composition, determining their applicability and estimated efficacy within the tasks obtained above.
- I implemented an application that may be trained on a set of music data (stored in the Midi format) and will provide an attempted continuation of a given musical input, according to the tasks obtained above.
- I designed and performed a series of objective and subjective tests to assess the performance of the application and the algorithms that comprise it.

Supporting Technologies

I used the Anaconda implementation of Python included with the Spyder IDE to create my application. Furthermore, I used a set of libraries implementing utilities and machine learning techniques to create the application. These libraries are:

- Mido for Midi file processing
- NumPy for efficient mathematical operations
- SciPy for clustering and regression
- PyBrain for neural networks
- HmmLearn for hidden Markov models

Acknowledgements

I would like to thank my supervisor Peter Flach for his guidance over the course of this project in both its technical aspects and in its effective management.

Chapter 1

Contextual Background

1.1 Overview

Algorithmic composition is an AI problem concerning the creation of music via some algorithmic process. It is currently one of the topics at the forefront of the greater study of *computational creativity*: the challenge of designing an AI that displays, according to some agreeable definition, creativity. This can be described in another way as algorithmically mimicking various aspects of human intelligence, particularly relating to the conception of novel solutions to problems. The arts are a particular subject of interest for this research, as they are widely considered to be highly creative and distinctly human.

From an AI researcher's perspective, algorithmic composition is not too different in some ways from most other AI tasks: the goal is to design an algorithm that can analyse some complex natural process, derive an abstract model from the available data, and then select actions according to that model. The subject being modelled in this instance however is a psychological process that is both difficult to measure and relatively poorly understood. Many attempts have been made to find an effective way to deal with this problem; some have attempted a bottom-up approach based directly on human psychology, some have attempted to build a top-down model based on known musical principles, and yet others have tried to tackle it as a purely data-oriented task.

1.2 Computational Creativity

Artificial intelligence as a field has grown a great deal from its first appearance on the academic landscape 60 years ago. It has suffered a fairly shaky history, rising and falling from prominence several times throughout its short life thus far, before reaching its current widespread mainstream adoption and high value in industrial application. Currently techniques that form a part of AI, such as machine learning (ML), are used in a huge number of environments to tackle a wide range of problems. It has become apparent that almost any area in which data analysis plays a large role is a viable target for AI intervention. This not only targets traditional data-heavy fields such as marketing or economics, but also enables complex tasks such as processing natural language, bioinformatics, and autonomous robotic locomotion. While growth and decline is not unknown to AI, it currently occupies a major role in the modern technological landscape and does not show any immediate signs of slowing.

Furthermore, the influence of AI spreads beyond the direct technological applications alone. Even since before the formalization of AI as a field, there has existed a cultural fascination with the concept of intelligent machines. There has been much speculation about the ethical and philosophical implications of the creation of an intelligent mind, and a great deal of fiction has been created with the topic as a central theme. In more recent times as AI has become widely prominent, new hopes and worries have begun to emerge with regards to the social impact, ethical impact, and potential dangers of advanced AI systems. It is in this context that the ultimate capabilities of modern AI technology has fallen into sharp focus.

Currently, the greatest weakness of AI is seen to be in displaying human-like creativity. Although AI is very powerful within domains that have been sufficiently well formalized, there are many fields that remain substantially informal. The most significant of these are the arts. Machines have, to this day, failed to demonstrate any artistic merit of significance that approaches the level of a skilled human. While AI has demonstrated a certain degree of proficiency at specific well-defined tasks within artistic subjects, such as identifying artists from their paintings [3] or genres of music [13], AI remains unable to perform broader creative tasks such as creating or appraising art in a human-like fashion. A good deal of research has been put towards tackling this identified weakness from the academic community, but despite occasional novel results overall progress is proving slow. This deficiency is exacerbated to no small degree by the fact that there is little overlap otherwise between AI and the arts, making it difficult for collaboration or the sharing of knowledge and results between the two.

1.3 Algorithmic Composition

One of the areas of artificial creativity currently under focus is algorithmic composition. Algorithmic composition is the general term for the application of algorithms in some form to the task of musical composition - in short, AI that can write music. This is a particularly enticing challenge for those interested in artificial creativity, as music is generally considered to be the most mathematical of all the arts. Various aspects of music at both the physical and compositional level exhibit a very strong degree of structure that can be described mathematically, and conversely elements of mathematics have been directly employed by composers in the creation of their work. This provides a strong argument for music as one of the best suited art forms for analysis by AI, as musical data may be meaningfully digested computationally and consists of patterns and structures that may apparently be learned and predicted.

In practice however, AI still has a long way to go. Much exploration has been made with the goal of finding an algorithm that produces satisfactory results, but as of yet even the most state of the art musical AIs have clear flaws and limitations. Most "successful" AI is specialized for a specific region of composition; common choices include harmonization [11], jazz improvisation [10], and full orchestral composition [8]. The best performers within their respective regions produce work that is subjectively assessed as approximately equal to that of a novice composer. Most such work contains noticeable idiosyncrasies however, and despite being able to create amateur-level music the AI consistently fails to produce work comparable with that of skilled composers at even a small probability. This is generally attributed by both artists and certain AI specialists as a lack of creativity; although the AI is skilled at applying certain musical concepts effectively it does not feature the combination of novelty and coherency common to human work. This is not to say that AI as we currently know it is not capable of composition at a human-like level, but that current methods fall short of the mark.

1.3.1 Current Challenges

One of the unique challenges in computational creativity, particularly in algorithmic composition, is that it is very difficult to define creativity to begin with. In some circles of research the goal of computational creativity is in fact to simply understand human creativity in the first place. Because of this, setting an easily measurable objective for the creation of creative AI is implausible if not impossible; instead the metrics used to measure success are typically focused on subjective assessment, either by direct comparison with existing creative work or appraisal by trained art critics. One fairly scientific approach to this is the "creative Turing test," a variant of the traditional Turing test in which a human subject converses through a terminal with both another human and a computer, without being informed which is which. The computer is considered to have passed the Turing test if the subject is unable to distinguish between the human and the computer through the conversation. This translates well into a test of creativity; a human may observe creative works produced by a computer and by a human, and attempt to distinguish which was created by the computer in the same manner. This can be a difficult test to pass depending on the particular creative medium being assessed. One important detail to note with

regards to this challenge however is that it does not assess the subjective quality of the piece. This is very understandable given that subjective quality is difficult to define and cannot currently be measured computationally, but it nonetheless is one of the most important factors for any practical application of a creative AI.

Beyond the more abstract difficulties involved however, there are significant technical challenges in replicating human-like musical composition. These include the compositional complexity of most musical pieces (even music considered relatively simple), the colossal variety of styles and genres of music in existence, and the difficulty in obtaining data relating to human appreciation of music. Most AI research thus far has primarily been concerned with creating a robust model for the complex patterns present in real music; typically the goal is either to imitate elements of a pre-specified corpus of music (as in [17] [24]) or produce wholly original work according to some prior-defined style ([8] [4] [14]).

Despite numerous attempts using a wide range of AI methods (described further in section 2.1) there have been few significant successes in achieving this goal. The most notable of these is Iamus, a supercomputer running software based on the Melomics system [8], and the first computer to compose professional contemporary classical music in its own style [1]. This system used an evolutionary algorithm with manually-encoded rules as a fitness function (detailed in 2.1.4), meaning that it was directly instructed on how to produce classical music without the need for any learning to take place. Iamus' work has received mixed critical reception, although there is the strong possibility of bias resulting from the fact that composer was already known by most critics to be a computer. Although this is a very impressive technical feat, it has not lead to any kind of industry adoption or significant public interest within the 6 years that have followed. This can be attributed to the fact that it exists as a standalone system; it does not integrate with any existing musical process and the work it produces is not in itself unique or useful.

1.4 Technology in Musical Composition

As detailed above, there has been a substantial amount of work invested into algorithmic composition up to this point. It is of notable importance that the vast majority of this work has developed from academic interest only, with very little direct involvement from industry towards the problem (and many of the companies that are involved emerged directly from academia). The cause for this is easily identifiable: there are few current financial incentives to pursue algorithmic composition. Although a high quality artificial composer would certainly be a valuable asset, technology is recognizably quite far away from achieving something effective enough to put to market with a high expected return on investment. This slows down the rate of progress in the study of algorithmic composition, and quite possibly computational creativity as a whole; while current research is going strong, the problem being tackled is immense and the resources available comparatively small.

The financial incentive for investment in algorithmic composition given the reasonable possibility of achieving success is also quite evident. The music industry is currently valued at over £10.0 billion globally [20], with the UK retail value worth over £900 million [2]. Any kind of new technology capable of providing a fundamental boon to production is therefore of high value.

In modern times, technology has served to greatly augment the compositional process. The invention of the analogue synthesizer and its subsequent rise to widespread popularity in the 1960s-1970s provided a powerful new tool to composers. This electronic instrument not only produced a unique sound unlike any traditional acoustic instruments, but proved highly versatile during composition due to the ability to synthesize sounds similar to different instruments. This functionality began to reach its true potential in the early 1980s, as polyphonic electronic keyboards incorporating many different sounds as part of their hardware began to enter popular use. This single piece of equipment gave individual producers the ability to both compose and perform each part of an entire multi-instrumental score by themselves. These advancements began to peak with the invention of software synthesizers and the MIDI format (see 2.4), imbuing a computer with the power to fully score and synthesize virtually any kind of music.

These advances have had enormous consequences for the world of music, as it has become possible for any individual with access to an inexpensive digital workstation to create, perform, and distribute music that would have required an entire orchestra or even been impossible to create otherwise.

1.5 AI in Creative Media

Although artificial intelligence has enjoyed enormous recent success in research and industry, it has also begun to see employment in entertainment as well. Although no AI has yet taken over a popular, mainstream creative role ordinarily performed by humans, there have been many cases where an AI has provided novelty or entertainment within an established art form.

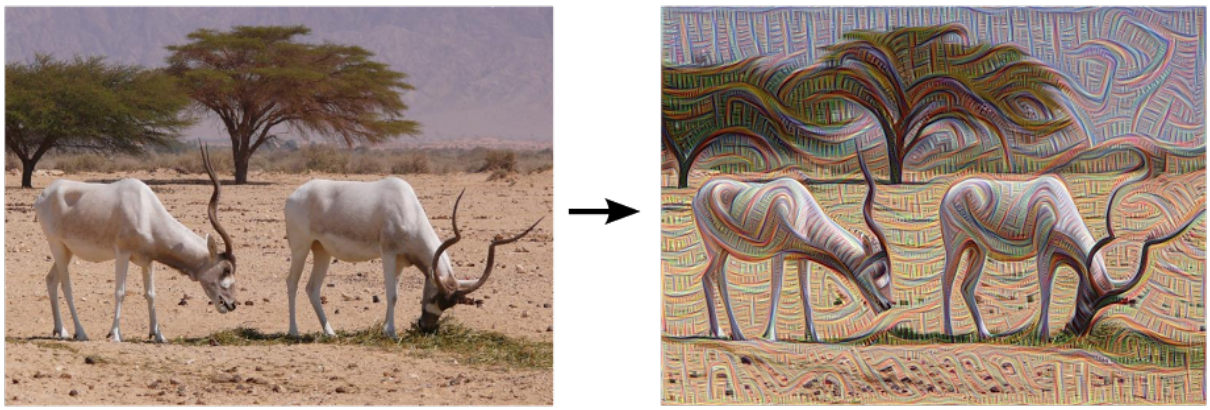

Figure 1.1: An image before and after processing by Google DeepDream

One of the most recent well-known cases of this is Google's "DeepDream," an online application that uses computer vision technology to enhance images according to detected features, morphing details to exaggerate falsely perceived visual elements (quite commonly biological) resulting in a highly surreal output reminiscent of dreams or hallucinations. This has received popular attention as potentially a form of true AI-created art, with numerous articles discussing the philosophical consideration that it could be considered true art [19] [12] and an actual art exhibition featuring its work [5].

AI has also demonstrated a surprisingly high proficiency in writing both fiction and non-fiction. Technically-oriented AI writers have already been employed in a professional environment by the company Narrative Science, in the form of their Quill platform: a service that analyses data, identifies data trends and other kinds of useful information, and performs natural language communication to relay results and answer questions [25]. Creative AI has also seen recent success as an algorithmically co-written story was recently entered into a literary competition (intended primarily for humans) and outperformed many of its human competitors [22]. In this sense, computer-human collaboration has already reached the level of a skilled human within the domain of writing.

1.6 Project Aims

The aim of this research project is to attempt to bridge some of the gap between the current theoretical state of algorithmic composition and the practical reality of AI in music. The final goal is the development of an algorithm that provides useful melodic material based on the work of a human composer in a manner that is realistic with regard to human composition.

To be considered successful, this algorithm must demonstrate the ability to generate monophonic melodic music in a style that is similar to that of an existing musical piece. As the goal in this case is not to simply create an entire self-contained song but to assist a human in the creation of their own song, the algorithm should not simply output a single complete melody. Instead, the algorithm must be able to provide a set of different outputs for each position within the song so that utility to the user

is maximized. Additional facilities to increase gain to the user, such as allowing the user to direct the generation process via some interactive means, are also being searched for. Finally, the program must be be such that regular use is feasible in a typical modern compositional context: there must be no hardware, time, or manpower requirements that would be extraordinary for a single musician working independently.

To summarise, the specific objectives of this project are:

1. Survey literature relating to algorithmic melody composition to determine a set of viable methods for expanding existing musical scores
2. Adapt one or more of these algorithms to write partial melodic sequences that fill gaps in or provide variations to an existing melody
3. Implement an application using this algorithm that allows operation by a non-technical user in a timespan appropriate for practical use (a reasonable estimation would be 1 minute per bar)
4. Assess the effectiveness of the algorithm(s) in the context of completing a melody through a set of objective tests
5. Determine, for each algorithm being tested, a strategy that enables user interaction with the melodic generation process in some guiding capacity
6. Assess the effectiveness of the extended algorithm(s) in the same manner as the original and determine the improvements offered by increased levels of human-computer interaction

Chapter 2

Technical Background

This chapter covers previous work in the field of algorithmic composition, both as a broad overview of the landscape and an in-depth explanation of the particular work directly relevant to this research.

2.1 Abstract Methods for Algorithmic Composition

Many common types of AI techniques have been used in some way for algorithmic composition. The majority of these techniques can be broadly categorised as one of:

- Knowledge-based models
- Supervised learning
- Evolutionary algorithms
- Hybrid models

Each of these categories carries their own set of advantages and disadvantages which greatly influence the type of output they produce. Although these categories are not restricted in the type of compositional tasks that they can perform, different techniques may result in completely different uses for the resulting AI. Examples of these different categories, some of the techniques that comprise them, and their strengths and weaknesses follow:

2.1.1 Knowledge-Based Models

Knowledge-based models use explicitly stored knowledge of some variety to perform computation. They are one of the more intuitive and easy-to-use types of AI, and represent the vast majority of early digital methods for algorithmic composition. There are generally two ways in which this knowledge is obtained; either it is explicitly added by the programmer when the AI is created, or it is learned implicitly from a set of sample cases. The former case has the clear weakness that it can only produce music that it has been explicitly taught how to produce; this puts a harsh limit on the versatility and creativity of the AI. Nonetheless this can be quite an appealing method to use, due to both its computational simplicity and the fact that it can be directly inscribed with compositional principles derived from well-established music theory. Implicitly obtained knowledge on the other hand is much more versatile but is heavily reliant on the quantity and quality of the data provided. The extraction of knowledge from a training set is also a difficult task with many possible solutions - this may also be validly categorised as supervised learning, but is listed here for simplicity's sake.

One of the main types of knowledge used in knowledge-based systems is a set of rules/constraints, which is combined with an algorithm that produces an output satisfying these constraints. These techniques are attractive as they have the most direct analogue to music theory. Due to their heavily restricted

form however, they are typically poorly suited to performing pure composition. Instead they are typically focused on broad but well-constrained tasks such as harmonization [23] or jazz improvisation over existing melodies [14]. Notably in recent decades there has been a significant shift from the use of hard satisfaction of constraints to probabilistic and optimizing methods.

Another common way to store knowledge is as a formal grammar. A formal grammar defines a kind of formal language consisting of a set of symbols that may be arranged according to strict hierarchical structural rules. By creating a grammar that uses musical elements of some kind as its basic symbols, the set of rules used by the grammar can be used to model a style of composition. This is useful when applied to music, due to the natural geometric structure behind most musical composition. However it can also be noted that formal grammars are limited in the degree to which they can comprehend long-term context, an important feature of music that formal grammars struggle to address [16].

2.1.2 Supervised Learning

Supervised learning is a subset of machine learning, in which the learning model is trained to approximate some function using a set of clearly defined example inputs and outputs. It is generally applied for tasks involving large-scale data analysis of some kind, such as classifying data samples or identifying relationships between variables in a dataset. It can also be applied to fairly complex decision-based tasks however, as in the case of algorithmic composition. Generally speaking, most supervised learning techniques used for algorithmic composition attempt to model music as a sequential pattern of notes, each associated with some corresponding output depending on the task (such as the subsequent note in melody composition or a harmony in harmonization). This can be considered a specific form of pattern prediction; by understanding the relationship between each set of notes played in a song, you can predict new notes based on a given piece. Due to the popularity and effectiveness of supervised learning in general, it is one of the more popular methods for algorithmic composition.

There are two particular classes of supervised learning techniques that are frequently used in algorithmic composition; the first of these is artificial neural networks (ANNs). ANNs attempt to model the actions of neurons within the human brain in their ability to learn arbitrary mappings between input and output domains. This is very useful for composition as network designs can easily be adapted to different compositional tasks or combined with another techniques (see 2.1.4). The majority of ANNs used in music are recurrent neural networks (RNNs), which have a form of "memory" allowing them to model sequences of inputs and understand context. Because of this RNNs are powerful tools for composing short melodies and harmonization. They do however still largely fall short at understanding long-term context and at forming coherent song structure. Additionally as in most ML algorithms, the performance of an ANN is heavily dependent on the musical representation used. Because the inputs to a network are strings of numerical (typically binary) values, the music must be translated into an appropriate format, which the ANN must then learn features from. Because of this a representation in which important features are distinctly visible will heavily outperform one in which those features are more obscured.

The other major class in supervised learning are Markov models. A Markov model is a model representing a stochastic process which transitions through a set of states, where the transition probability is dependent only on the current state (i.e. there is no context or memory). The simplest of these is the Markov chain, which simply acts as a state machine where each state emits a fixed output when arrived at and then probabilistically transitions to another based only on the current state. At first glance this appears to be a crippling weakness, as music is intrinsically involves building on a history of musical events; there are a few ways in which this is resolved. Such methods include n-order Markov chains, which have memory of the previous n states, or the combination of a Markov chain with other processes. These methods have a limited effectiveness, but still display some stark idiosyncrasies that feel "out of place" in a tune. More commonly in recent years however is the use of the hidden Markov model (HMM), in which the exact state of the model at any given time is unknown, and each state outputs values with a certain probability. Training this model involves optimizing both the transition and output probabilities to fit the entire length of the training set, resulting in a much more sophisticated overall model. As a

result HMMs see much more use in composition, primarily in non-melodic tasks such as harmonization or rhythm.

2.1.3 Evolutionary Algorithms

Evolutionary algorithms (EAs) are a relatively recent addition to the algorithmic composition landscape, with the first cases appearing in the early 90s but the majority of practical attempts being made post-2000. EAs are an interesting class of problem solver in that they do not directly solve the problems they are applied to, but instead are a kind of optimization algorithm. One of the keys to an EA is the fitness function it is given; the goal of the EA is to find an optimal input for this function. In the context of music, defining a good fitness function is in itself a very difficult challenge - doing so would mean being able to objectively determine the quality of a musical piece, which is already a major component of the problem of algorithmic composition. Because of this it is quite common to combine EAs with other techniques for use as a fitness function in some capacity. Two of the most common examples of this are training ANNs to act as fitness functions on

2.1.4 Hybrid Systems

2.2 Artificial Neural Networks in Algorithmic Composition

As described in section 2.1.2, artificial neural networks (ANNs) are a powerful class of techniques for approximating functions, and recurrent neural networks (RNNs) extend their functionality to add a form of memory allowing for the comprehension of sequential data. Within this class there have been many different techniques created for performing melodic composition, with many featuring unusual novelties and useful unique properties.

2.2.1 Jordan Network With Sparse Pitch Representation and Planning Units

The first well-documented attempt at using ANNs for musical composition was made by Peter Todd [24], who created a simple 3-layered RNN with a topology based on a Jordan network [15] which composed monophonic melodies.

The representation of musical data used by the network is quite basic and has limited scope. The temporal dimension of a song is represented by dividing it into a set of discrete time slices at fixed intervals, and at each of these time slices exactly 1 note may be played or sustained. The input string to the network represents 1 time slice, and consists of a set of values representing pitches, 1 value representing the start of a new note, and a set of values corresponding to "plans." The pitch value for a given time slice is represented simply by taking the pitch of the note p within some user-defined scale (Todd used the C major scale with $p = 0$ for D4 up to $p = 14$ for C6), then setting the p-th pitch input to 1 and all others to 0. This only allows a fairly limited range of notes to be played, which reduces the versatility of the network. The value of the "new note" input is 1 if a note was started during the time slice, and 0 otherwise. Finally, the "plan" inputs have their values set according to the song; there is 1 plan input for each song in the training set, and that input is set to 1 while training with that song (with all other plan inputs set to 0). This input is used by the network to differentiate between the songs that it is trained with, allowing it to behave differently depending on the intended output. Finally, the output of the network is of the same format as the input sans the plan units, and represents the network's estimate for the next time slice.

This particular type of Jordan network is similar to a basic feed forward ANN, with an input layer, a hidden layer, and an output layer, with the input and hidden layers fully connected to the layer ahead. In addition to these connections, two recurrent connections are added: a connection from each i-th output neuron to the i-th input neuron with weight 1, and a connection from each input neuron to itself (with the exception of the plan neurons) with weight $\alpha < 1.0$ (Todd used $\alpha = 0.7$). This effectively treats the network's output as its input for the next iteration, and causes its inputs to slowly decay instead of

disappearing with each iteration. This gives the network an effective memory of recent notes that have been played, with each pitch fading out of memory the longer it goes without being played.

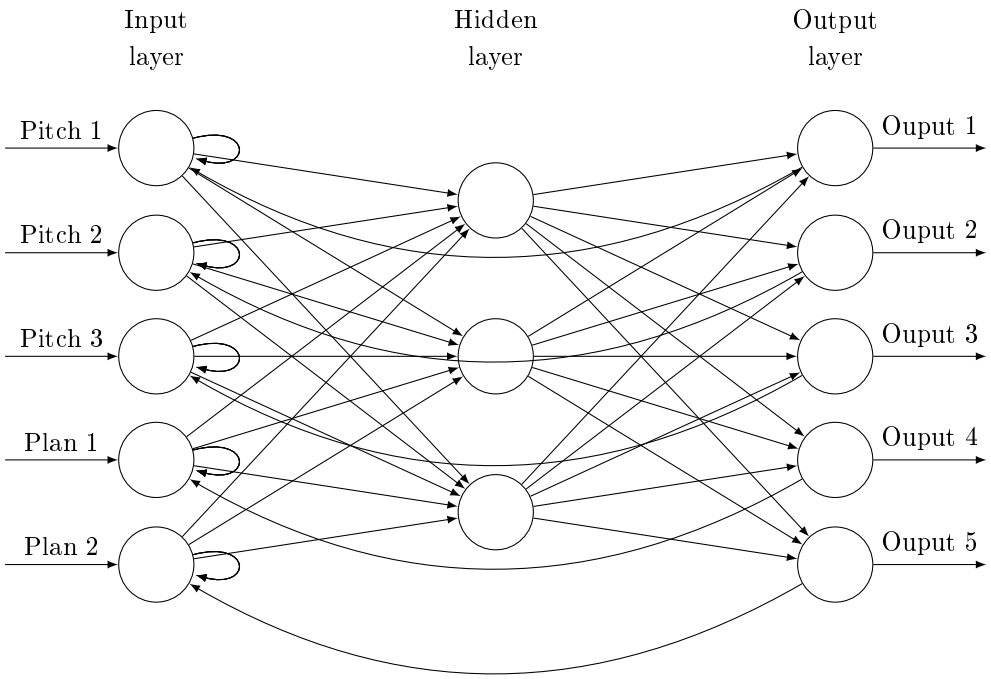

Figure 2.1: An example of a variety of Jordan network

The network is trained using standard backward propagation of errors with gradient descent. The training data is extracted from a set of songs by converting each song to the time slice format as described above, and providing them as input to the network. It is important to note that in traditional operation of the network, the only input to the input layer comes from the recurrent connections from the input layer and the output layer. However this leads to decreased performance during training, due to the fact that the input layer is receiving values from the output layer that are different to the expected output at that stage. Put simply, this means that instead of learning to map each time slice in the training song to the following time slice, the network learns to map the output from its current untrained state to the following time slice in the original song. This problem disappears as the network begins to converge and its actual output begins to match the expected output, but this can take a great deal of time as the training process is essentially chasing a moving target - the network is repeatedly trained to learn an incorrect mapping until it has converged enough to start learning the correct mapping. To overcome this problem, the training process eschews the recurrent connection from the output to input layers and instead directly provides the time slices from the training song as input, allowing the network to start learning the "real" mapping immediately.

This network was found to perform reasonably well by Todd. The primary testing metric used in the original paper was the computation required for the network to learn to recreate a number of distinct melodies. The results were fairly promising for the time, with a network of hidden layer size 15 being sufficient to learn up to 4 different songs of 34 time steps each. The number of epochs of training required notably scaled linearly with the number of songs; 2,000 for 1 song, 4,500 for 2 songs, and 8,500 for 4 songs. Reducing the number of hidden units from 15 to 8 however increased the number of epochs required to learn 2 songs to 50,000, due to the increased difficulty of representing the complex patterns of the song with a simpler function.

Following this, a subjective assessment was formed on the ability of the network to create new melodies

outside of the training set. This displays one of the advantages of using planning units, as by enabling or disabling different plan inputs the network can be made to output songs of different styles drawing inspiration from one or more of the training songs. A number of observations were made about the original compositions produced by the network: one interesting property it displayed was that when using a single plan unit it tended to output melodies consisting of strings of pitches from the corresponding training melody, but stitched together in new orders at points where the pitches aligned. This ultimately results in the melodies eventually beginning to loop, as at some point the network ends up in a similar state to its initial state and starts playing the opening sequence again. It was also observed to have a very weak understanding of rhythm - the single neuron for representing the start of new notes was insufficiently complex to model the structured nature of musical rhythm. Despite training, the network was unable to establish a relationship between pitch changes and note beginnings.

2.2.2 Elman Network with Circle-of-Thirds Pitch and Chaotic Inspiration

An alternative simple RNN to the Jordan network as described above is the Elman network [9]. The topology of an Elman network is similar to that of a Jordan network, but with different recurrent connections. While in the Jordan network the recurrent feedback is received in the input layer, in the Elman network it is instead received in the hidden layer. The specific model of Elman network for composition described here is as designed by Coca, Romero, and Zhao [6].

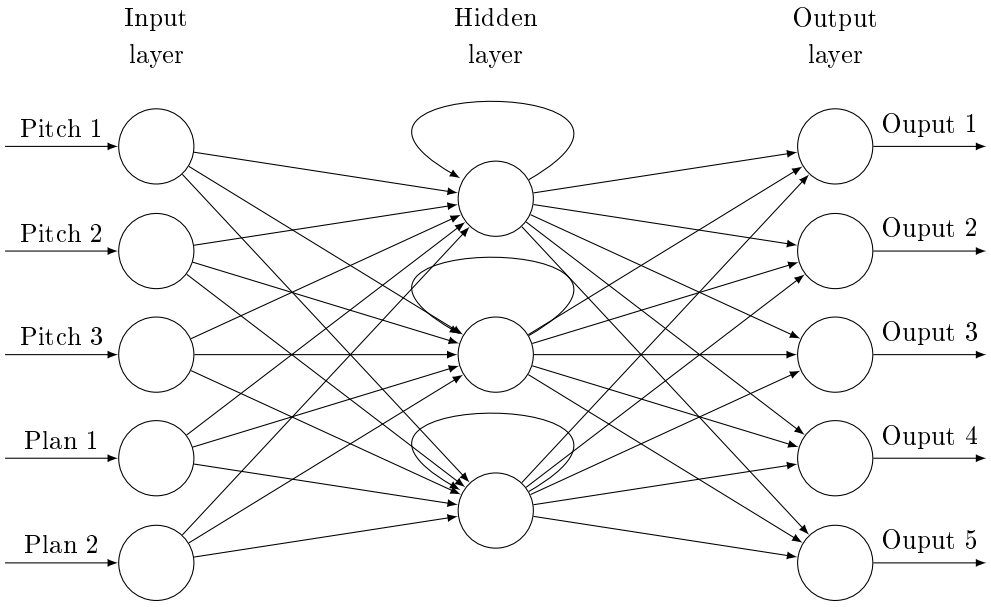

Figure 2.2: An example of a variety of Elman network

The representation of music used by this particular network is quite different to that used in 2.2.1. Most distinctly it does not use the time slice representation for the temporal dimension, but instead assigns a duration value to each event. This shares the disadvantage of only allowing monophonic input/output, but places a greater emphasis on learning the temporal features of the input data. This helps overcome some of the issues mentioned earlier with regard to the "new note" feature. The representation for the duration of a note is given as a binary string whose actual binary value represents the duration of the note. The conversion between numerical value and musical time is such that a crotchet (quarter note) has a duration value of 96. This is the exact value used by the MIDI standard (2.4), chosen such that it can represent all standard note divisions (semiquavers, crotchet triplets and quaver triplets) as well as the small irregularities in timing that occur during live performance.

The pitch representation is also quite different, as there is no longer a one-to-one mapping of pitches to nodes. Instead, the pitch is split into two parts: the in-octave pitch and the octave. The representation for the octave is very simple and uses only 2 nodes, with the binary mapping 10 = C2-B2, 01 = C4-B4, and 00 = C3-B3, with the network's output being snapped to the closest of these values. This can easily be extended to cover a wider range with a small increase in nodes, making this representation more versatile than the one used in Todd's design. The representation for the in-octave pitch is also split into two parts, based on intervals of 4 and 3 semitones (major and minor thirds). Each circle represents a set of pitches whose relative intervals are multiples of 4 and 3 for major and minor thirds respectively. A given note is then represented as a pair of binary strings length 3 and 4 representing the index of the circles it lies in. So for example, C is encoded as 1000 100, D is encoded as 0010 001, and E is encoded as 1000 010. This is a useful representation as it models relative pitch (for certain intervals), which is more relevant in most musical structure than absolute pitch, without some of the disadvantages imposed by purely relative pitch representation such as easily straying off-key.

Chaotic Inspiration

The network also features an additional set of input nodes used for chaotic inspiration based on the algorithm described in [7]. This algorithm is based on the principle of generating music through chaotic dynamical systems. This is a branch of algorithmic composition, first explored in the late 1980s [18], that differs greatly from the AI methods previously explained; instead of attempting to directly model compositional methods in some way, a mathematical model is developed that maps systems with chaotic behaviour to musical scores. The power of these systems comes from the fact that these models generate fractal objects with structures similar to those found in the analysis of classical music. Naturally most results in this area are not well suited to composing complete musical scores due to their unrefined and somewhat erratic nature, but are quite well suited to producing musical material from which a composer can be inspired.

Coca, Tost, and Zhao's algorithm uses a nonlinear dynamic system consisting of three variables. These variables are drawn from an existing melody, with the first being extracted from pitches, the second from rhythm, and the third from musical dynamics (amplitude and velocity). It also takes a set of inputs defining a specification for the musical scale in which the output piece is to be written. The algorithm then uses these inputs to map the extracted variables into a new melody, stored in the MIDI file format (2.4). As is expected from a chaotic system, a small change to the input variables can cause a disproportionately large change to the output values while still producing a result of similar quality.

This algorithm is used to generate a source of "inspiration" that can be used in the creation of a melody. The goal of inspiration in the context of algorithmic composition is to create variations in the melody generation process by modifying the system in a way that preserves the original information; typically this is accomplished by adding a new input to the system which provides a source of variance. This is very useful for systems in which a large number of possible outputs are desired, such as those that present their results to some selection process (e.g. evolutionary algorithms). The use of chaotic algorithms to generate this inspiration is advantageous due to the fact that their output has a similar fractal structure to real music, meaning that a given inspirational segment contains a structure that the final output melody will be built on in some way.

In this particular compositional algorithm, the input melody from which the chaotic inspiration is drawn is the same as the melody being learned during training, or the melody being recreated while composing. A subset of the notes from the chaotic inspiration are provided as input to the network, . In the paper ([6]) the varied results are provided by changing the number of notes

2.3 Distance Modelling with Hidden Markov Models for Monophonic Rhythms

As discussed in section 2.1.2, Markov models tend to have trouble with the kind of complex structure involved in melodic pitch sequences. Using a hidden Markov model helps improve the model's performance in dealing with more complicated sequential patterns. While this type of model is still quite weak in terms of comprehending pitch, it is more capable of recording and recreating simple rhythmic patterns across moderate durations. While this alone is not a common use for HMMs due to their inability to model true long-term patterns, they can be used as part of a hybrid system using a model of rhythmic distances to greater effect [17].

The HMM used to model rhythmic patterns uses a fairly straightforward time slice representation for rhythm. The time of

2.4 The Midi Format

Chapter 3

Project Execution

3.1 Design

3.1.1 Composition Algorithm

One of the key details that sets the intended method apart from most other algorithmic composition methods is its purely cooperative nature. This allows for a major performance benefit by offloading a certain portion the work to the human operator. There are three major places in which this offloading can take place: initializing state before generation, guiding the generation process, and adjusting the final generated output. The first is achieved by using the score written by the user as input to create new material from. The third is an intrinsic part of the process, as the final output of the algorithm is provided to the user to be used as they please. The second may be exploited to improve the performance of the algorithm in generating output that is useful or pleasing to the user.

There are a number of ways that this effect could be achieved. One would be the use of evolutionary algorithms. Due to their use of a fitness function to guide their search through the solution space, it is simple and straightforward to integrate user feedback into the EA by using user rating - either as the fitness function, or as some kind of augmentation to the fitness assignment process. A number of EAs have been designed with this behaviour in mind in order to overcome the challenge of defining an objective fitness function. This does however suffer a major drawback in that it is not feasible for an individual user to guide the search process single-handedly; due to the large number of samples that must be judged, user fatigue rapidly becomes a critical constraint. There have been attempts to mitigate this effect through various means, such as reducing the population, number of generations, or novel methods such as using clustering to avoid presenting the user with similar outputs. These methods each present their own limitations however, to an extent that was prohibitive in the case of this research project.

3.1.2 Training

As discussed in sections **??** and **??**, the AI contains several components using supervised learning. Therefore the results produced by these components, and by extension the AI, will be heavily dependent on the training data provided.

Providing training data for the AI to learn from is potentially a non-trivial task. The reason for this is that the data must fulfil a certain set of constraints, namely:

- The data must be stored in a machine-readable format
- The input must not contain any features that the AI is not capable of parsing
- There must be a strong degree of consistency within the data so that the learned patterns are coherent

The format used to store music throughout this project is the Midi file format [21], which stores music in the form of notes and events instead of sound. This is ideal for this application, as it allows for direct recovery of the exact musical elements contained within each piece of music, a significantly more challenging task when dealing with raw audio. It is also convenient for data acquisition as there exist various large collections of royalty-free midis available to the public. Unfortunately these midis are very often classical music of some variety which often fails to satisfy the second requirement.

The second requirement is one of the most constricting, as for various reasons the AI has many restrictions on the type of musical data it can work with. The most harmful of these is the inability to parse or generate polyphonic melodies. This causes some major issues when working with orchestral or piano music due to their frequent use of many melodic lines and chords respectively, which excludes most available classical music. Although it is in some cases possible to isolate a single melodic line, there are quite often no such lines that are both long and substantial enough to be useful for training.

To best overcome these restrictions, the data used was a selection of nursery rhymes transcribed for this research. These have the advantage of being relatively simple and sharing many commonalities such as key and meter, making it much easier for the AI to identify and recreate patterns. They are also universally monophonic due to their vocal nature.

3.2 Implementation

Bibliography

[1] Computer composer honours turing's centenary. *New Scientist*, 2012.

[2] Riaj yearbook 2015: Ifpi 2013, 2014. global sales of recorded music. Technical report, Recording Industry Association of Japan, 2015.

[3] Alexander Blessing and Kai Wen. Using machine learning for identification of art paintings. Technical report, Stanford University, 2010.

[4] Anthony Richard Burton. *A hybrid neuro-genetic pattern evolution system applied to musical composition.* PhD thesis.

[5] Kelsey Campbell-Dollaghan. Inside google's first deepdream art show. *Co.Design*, 2016.

[6] Andrés E Coca, Roseli AF Romero, and Liang Zhao. Generation of composed musical structures through recurrent neural networks based on chaotic inspiration. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, pages 3220–3226. IEEE, 2011.

[7] Andres E Coca, Gerard O Tost, and Liang Zhao. Characterizing chaotic melodies in automatic music composition. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 20(3):033125, 2010.

[8] Gustavo Diaz-Jerez. Composing with melomics: Delving into the computational world for musical inspiration. *Leonardo Music Journal*, 21:13–14, 2011.

[9] Jeffrey L Elman. Finding structure in time. *Cognitive science*, 14(2):179–211, 1990.

[10] Judy A Franklin. Multi-phase learning for jazz improvisation and interaction. In *Proceedings of the Eighth Biennial Symposium for Arts & Technology*, 2001.

[11] Alan Freitas and Frederico Guimaraes. Melody harmonization in evolutionary music using multiobjective genetic algorithms. In *Proceedings of the Sound and Music Computing Conference*, 2011.

[12] Marina Galperina. Is google's deep dream art? *Hopes & Fears*, 2015.

[13] Michael Haggblade, Yang Hong, and Kenny Kao. Music genre classification.

[14] Damon Horowitz. Representing musical knowledge in a jazz improvisation system. In *Proceedings of Artificial Intelligence and Music, IJCAI workshop*, pages 16–23. Citeseer, 1995.

[15] Michael I Jordan. Serial order: A parallel distributed processing approach. *Advances in psychology*, 121:471–495, 1997.

[16] James Anderson Moorer. Music and computer composition. *Communications of the ACM*, 15(2):104–113, 1972.

[17] Jean-François Paiement, Yves Grandvalet, Samy Bengio, and Douglas Eck. A generative model for rhythms. In *NIPS Workshop on Brain, Music and Cognition*, number LIDIAP-CONF-2007-035, 2007.

[18] Jeff Pressing. Nonlinear maps as generators of musical design. *Computer Music Journal*, 12(2):35–46, 1988.

[19] Alex Rayner. Can googleâĂŹs deep dream become an art machine? *The Guardian*, 2016.

[20] Richard Smirke. Global record business dips slightly, u.s. ticks upwards in ifpi's 2015 report. report, billboard.com, 2015.

[21] Andrew Swift. A brief introduction to midi. *URL http://www. doc. ic. ac. uk/˜ nd/surprise_ 97/journal/vol1/aps2*, 6, 1997.

[22] Andrew Tarantola. Ai-written novel passes first round of a literary competition. *Marketing Land*, 2016.

[23] Marilyn T Thomas. Vivace: A rule based {AI} system for composition. 1985.

[24] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4):27–43, 1989.

[25] Alex Woodie. Your big data will read to you now. *datanami*, 2014.

Appendix A

An Example Appendix

Content which is not central to, but may enhance the dissertation can be included in one or more appendices; examples include, but are not limited to

- lengthy mathematical proofs, numerical or graphical results which are summarised in the main body,
- sample or example calculations, and
- results of user studies or questionnaires.

Note that in line with most research conferences, the marking panel is not obliged to read such appendices.