Literature Review and Project Specification: Algorithmic Generation of Novel House Music

Frederick Glass

21st November, 2018

Abstract

An investigation into the computational techniques for artificially composing music, namely neural networks and evolutionary computation. For each topic, extensive analysis has been undertaken on its historical background, relevant theories and concepts, current research scope and its application to music generation. A project on the algorithmic composition of innovative house music is then proposed, with detailed sections specifying the requirements, research hypotheses and evaluation criteria. This project strives to bridge the gap between algorithmic composition and the electronic dance music world, where there seems to currently be a distinct lack of academic research available, and hopefully inspire other studies of a similar nature to be initiated.

I certify that all material in this dissertation which is not my own work has been identified

1 Introduction

House music is a sub-genre of Electronic Dance Music (EDM) and was born in Chicago [1], with heavy influence from DJ Frankie Knuckles as well as the introduction of the drum machine, in particular the Roland 303. This compelling uprising of the drum machine indicates a robust relationship between music and technology; one which is aimed to be taken even further in this project, through the minimisation of human intervention and the addition of artificial intelligence [2]. A typical house track has a structure which involves an intro, a chorus, various verse sections, a mid-section and an outro. It also has a tempo which lies between 118 and 135 beats per minute (bpm), with the sweet spot often considered to be 128 bpm.

There are two divergent forms of house, vocal and non-vocal [3]. The latter shall be the primary focus of the project, as the lack of vocals signifies no voice synthesis is required – a standalone task in itself. Furthermore, the drum machine has a more prominent influence in this type, causing the rhythm, which follows a four-on-the-floor pattern, to be a track's integral element. This rhythm generally has a constant tempo, lying between 118 and 135 beats per minute (bpm), with the *sweet spot* often considered to be 128 bpm. This repetitive tempo makes house music an ideal candidate for algorithmic composition due to its regularity and lack of variation. As a result of this, short generated fragments can often be relatively easily extended with minimal effort. Additionally, house is entirely digitally produced and therefore does not suffer from a distortion in results, due to the absence of noise [4].

It is also worth noting that this project is targeting the generation of *style imitations* [5], where this is the attempt to reproduce music in the style of an already established genre – in this case, house. This is as opposed to *genuine composition*, where compositional techniques are used to generate entirely original pieces of musical art; something which is far more difficult and has significantly less academic research on. As a result, any further reference to algorithmic music composition indicates style imitation. Leading on from this, it is important to clarify that, in this context, *novel* compositions merely imply that they are different from the input material, in contrast to entirely new pieces.

The computational generation of music is not a new field, with genres such as blues [6], classical [7, 8] and rock [4], amongst others, already being successfully synthesised. However, considerably less research has been done on EDM, with almost none on house, resulting in a challenging lacuna which this project aims to help fill. There is a plethora of methods for algorithmic composition, including, but not limited to, Markov models, generative grammars, transition networks, chaos theory and cellular automata [5]. However, this review shall concentrate on another two; neural networks [9] and evolutionary computation [10], which have both proved effective in generating novel music.

Neural networks shall be explored first, with respect to their structure, the variations that exist, how they are applicable to music and their feasibility for being applied to this problem area; the encoding and representation of its input and output is also examined. Next, the field of evolutionary computation shall be delved into, paying attention to similar areas as with neural networks. The technologies which can be used to undertake the actual implementation shall also be briefly reviewed. Finally, ideas from the various resources used will be combined to form a critical analysis section, which will conclude the exhaustive literature review. After that, the extensive project specification shall be defined in order to provide a complete and thorough document on artificial house music generation.

2 Literature Review

This section shall contain the bulk of the paper and will consist of research into the subject area as well as the various computational technologies that will be applied to that area. Critical analysis of this research shall also be undertaken and will finish the section.

2.1 Neural Networks

Neural networks are a relaxed representation of the neural structure found in a biological brain and were first proposed by McCulloch and Pitts in 1943 [9], where a simple one was modelled using electric circuits. Since their introduction, Artificial Neural Networks (ANNs) have had a substantial amount of research done on them, greatly progressing as a result. They have now been applied to an array of use cases, such as for machine translation [11], pattern recognition [12] and even medical diagnosis [13].

2.1.1 Structure

In a basic network, a single node takes a set of inputs, known as the *input layer*, and performs a weighted sum of these values, i.e., multiplying each one by some weight before taking the overall sum. Next, a constant, known as *bias*, is added and the overall sum is then clamped using a nonlinear activation function [6], such as a sigmoid function, with the exact one chosen depending on the type of learning method used.

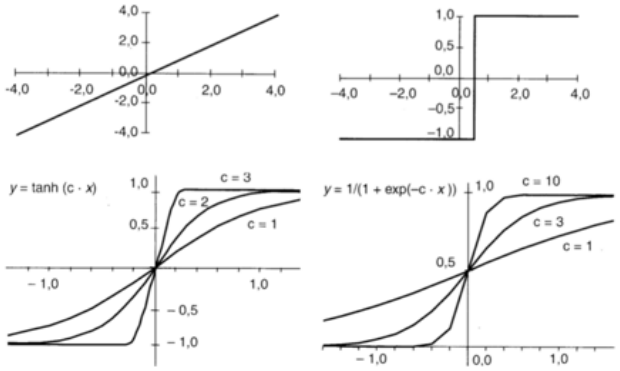

Figure 1: Different activation functions of an ANN [14]

Each network has multiple nodes, which are all fed identical inputs but can have different weights and biases, arranged into *layers*. The final layer of nodes is called the *output* layer, with any transitional layers between this and the input layer being referred to as *hidden* layers. If one or more hidden layers exist, the ANN can be referred to as a deep neural network [15] and is therein categorised within the deep learning field, a subdivision of machine learning [16].

Given an input, a loss function (also known as a cost function) is applied to determine how good or bad any given output is, i.e., the net error is calculated, which is the difference between the expected and actual output. Once this cost value has been obtained, *backpropagation* [17, 18] can be used. This essentially calculates the gradient of the cost, with respect to the weights, and then uses some optimisation method, such as Hessian-free optimisation [19], to update the connection weights accordingly, in order to minimise the cost [5]. Another training technique is backpropagation *through time* (BPTT) [20], an iterative gradient descent algorithm more suited to certain neural network architectures.

Neural networks have the ability generalise [21], meaning that they perform just as well on new, unseen data as they do on the data they were trained on. In order to achieve a high degree of generalisation, datasets should be segregated into the following parts [22]:

- A training set, used to train the network
- A validation set, used for gauging the performance on patterns that have not been seen and for tuning the model parameters
- A test set, for overall performance evaluation

2.1.2 Variations

Numerous versions of a typical ANN exist, with types varying by either their architecture or by the learning function they use. *Feed-forward* neural networks (FFNNs) [15] are the original ANN and can be single-layer or multi-layer perceptron's, depending on whether any hidden layers exist; their structure is as detailed in the preceding subsection and Figure 2.

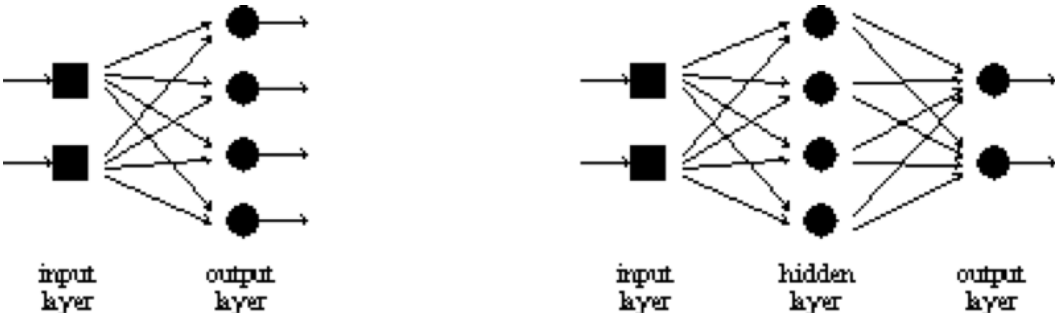

Figure 2: The structure of single-layer (left) and multi-layer (right) FFNNs

In these networks, information only flows in a single direction: input to output. With *Recurrent* Neural Networks (RNNs) [15], this directional constraint does not exist; instead, the output of each hidden layer can be delivered back to itself as an additional input, allowing variable-length input and output. This feature introduces the concept of *memory* to an RNN, allowing it to take advantage of its internal state with negligible overhead, making it suitable for tasks like speech [23] and handwriting recognition [24, 25].

However, one problem of a regular RNN is that its memory is very short because values are lost if they are not passed down in the subsequent time step. To combat this, a Long Short-Term Memory (LSTM) [26] neural network can be used instead, a variant of an RNN which establishes units utilising a *memory value*. This value is passed down multiple steps, in parallel with the activation output, and can be added or subtracted at each one, providing the network with a means to greatly improve its memory capabilities. This enhanced memory allows the network to learn long-term dependencies of the data, hence solving the *vanishing gradient* problem [27] by imposing constant error flow. LSTM units also solve the *exploding gradient* problem, where error gradients amass and result in sizable updates to weights during training, by having a linear structure which has a fixed self-connection and is encompassed by non-linear gating units in charge of controlling information flow [6].

Another alternative to a LSTM RNN is one which has Gated Recurrent Units (GRUs), presented by Cho et al. in 2014 [14], that also intend to solve the vanishing gradient problem. They have fewer parameters than a LSTM unit, as they do not consist of an output gate, but with similar performance gains when compared to traditional tanh units.

2.1.3 Application to Music

Amongst their other operations, ANNs have been applied to music in an abundant number of use cases and studies. The scope of application is fairly broad, with neural networks even being used in the field of classification. For instance, Dannenberg et al. [28] utilised a Cascade-Correlation architecture [29] to construct classifiers helpful in style recognition. Kiernan [30] also capitalised on the power of ANNs for score-based style recognition.

Recurrent neural networks are particularly suitable for algorithmic music generation. Hild et al. [7] used three parallel recurrent networks, each with a hidden layer, to harmonise melodies in the style of Bach. Classical music is a ubiquitous genre in music composition with Huang and Wu [8] using a two-layer LSTM RNN to learn the relevant music structures and synthesise it. Nonetheless, other genres do receive plenty of attention. For example, original blues music was algorithmically

composed by Eck and Schmidhuber [6]. This was done using a LSTM RNN again, signifying just how adept this architecture can be at learning the local and global music structure and generating novel sounds.

Johnson [31] took a slightly different approach, by using an RNN inspired by the architecture of *convolutional* neural networks (CNNs) [32]. A CNN can search for shapes across an entire image with minimal learnable variables, making them proficient at feature detection in image recognition. This architecture combination did not manage to maintain style over extended periods of time, but it was able to model measure-level structure.

As previously mentioned, GRU RNNs can have similar effectiveness compared to LSTM RNNs. However, this is not always the case when applied to music composition, and is largely dependent on the input encoding, with Nayebi and Vitelli [4] finding that they are actually remarkably less musically plausible due to a lack in representational capabilities. In this academic study, LSTM networks were trained using the raw audio waveforms of rock compositions from David Bowie and songs produced by Madeon, an EDM artist specialising in electropop, nu-disco and house genres. Music in this style was then successfully algorithmically generated, proving that it is possible to artificially compose house music, as this project also intends to do so, using the aforementioned neural network topology. This paper also made use of spectrograms (Figure 3), to help visualise the network and uncover what it was actually learning. This was done through plotting the mean magnitudes of validation set sequences against frequency for both the LSTM and GRU networks, and then comparing these visualisations to a ground truth mean spectrogram. It found that the GRU network had greater activation bands, which is not a desired characteristic and likely accounted for its poorer compositions, due to the higher levels of noise present.

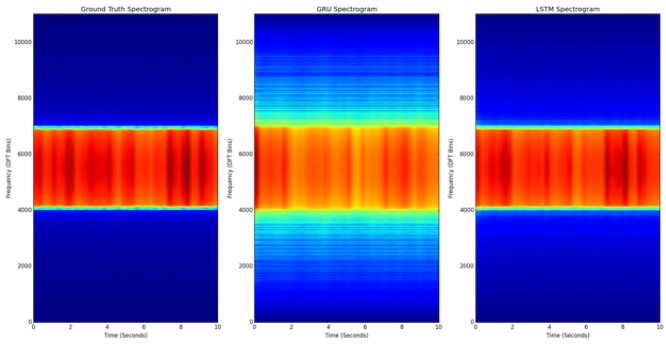

Figure 3: Comparison of spectrograms [4]

2.1.4 Encoding and Representation

An appropriate encoding scheme is vital to ensure the neural network can effectively interpret the input and output values, as well as being liable for the implementation of the correct learning algorithm. Firstly, it is important to highlight the difference between *encoding* and *representation*; the former is a way "to change information into a form that can be processed by a computer", whilst the latter is "the act of presenting somebody/something in a particular way" [33]. Sometimes these terms are used interchangeably but, in this context, encoding is a prerequisite for processing input musical data, whereas representation is about displaying the musical information in some fashion, e.g., mapping the output onto note values in different manners [5]. In essence, the representation is accountable for supplying the encoding with the information that *optimally* describes the musical material; unsurprisingly, this is highly coupled with the nature of music being represented.

The representation is crucial in any system and can cause a plethora of issues if not undertaken in a pertinent manner; for example, a *relative* representation [5] can cause a net error to propagate in neural networks. An *absolute* representation is therefore often favoured, especially within the

scope of neural networks, and is further specified as a *localist* [34] or *distributed* [35] representation. For example, in regard to pitch, a localist representation would have each pitch allocated to a particular output neuron. On the other hand, in a distributed representation, each pitch would have multiple output neurons assigned to it; this is not an ideal characteristic and may well cause the ANN to be difficult to train [36]. A simple practical example to summarise encoding and representation is that of degrees of a tempered scale, which can be encoded as binary or decimal values and represented in an absolute way where specific values correspond to specific notes.

Along with directly manipulating the raw waveform representation of the musical data [4], input can be defined in a collection of other means. Musical Instrument Digital Interface (MIDI) [37] is a medium for sending digitally encoded music material between electronic devices. Martens and Stuskever [19] converted a large number of MIDI files into *piano-roll*, a series of binary vectors describing a batch of notes at the present time, for use in their RNN. Conversely, Mozer [34] used a multi-dimensional representation model that incorporates multiple musical components like harmony, rhythm and pitch.

2.2 Evolutionary Computation

Evolutionary computation [10] is a subsidiary of artificial intelligence, encompassing optimisation algorithms motivated by Darwin's theory of evolution [38]. This project shall concentrate on *Genetic Algorithms* (GAs), initially introduced by John Holland [39], which are a class of evolutionary algorithms. They are stochastic search methods heavily influenced by the process of natural selection and are affluent within many diverse computer science domains, from biological sequence analysis [40] to optimising electromagnets [41].

2.2.1 Structure

Genetic algorithms have a relatively generic structure template, and no domain-specific knowledge is required to actually apply it. First off, a population of random individuals, or chromosomes, is generated. This population is then subjected to the following iterative process, as described by Goldberg [42]:

1. Evaluate the fitness of each chromosome
2. Apply a selection method
3. Apply the genetic operators
4. Create a new population and return to step 1 to repeat the process, provided the termination criteria has not been met

Figure 4: Flow chart representation of a standard genetic algorithm

Fitness is evaluated according to a *fitness function*, which is specific to the problem at hand. For example, the sudoku solver constructed by Weiss [43] assesses the fitness by calculating the number of matching symbols in the rows or columns, with the assumption that fewer duplicates implies a superior solution. Fitness functions may be mathematical, represent a comparison set or a rule-based system, or even be human-judged [5]. Sims [44] used this human evaluation approach and allowed users to select their favourite images, which then went on to become parents for the next generation. Latham and Todd [45] also used a similar mechanism, but with a single image constraint.

Selection represents the strategy for determining which regions of the search space should be probed, with a number of selection methods existing [46]. Fitness proportionate, or roulette wheel, selection is one such method, where the probability of selecting an individual i is:

$$\frac{f_i}{\sum_{k=1}^{P} f_k}$$

This is equivalent to spinning a roulette wheel, hence its namesake, with sectors proportional to fitness. However, this method is not recommended for use as it has problems, unless special considerations are taken, when:

- Trying to minimise the fitness
- The fitness is negative
- A fitness is large relative to the rest of set

Rank-based selection is where the selection probabilities are proportional to rank, with the fitnesses being ranked from population size, fittest, to one, least fit. The issue with this approach is that computing the rank requires sorting, which has time complexity $O(n \log n)$; an expensive operation for large datasets. Another selection procedure is *tournament* selection, which avoids the problems of the previous methods, but requires an additional parameter, parameter size, to tune.

Genetic operators [47] offer a way to generate new candidate solutions, through exploitation and exploration [48]. Exploitation involves having operators which have a strong chance of yielding good new solutions, through modest changes, and exploration is concerned with discovering as much of the solution space as possible; ideally, there will exist a balance between these two factors. There are two principal genetic operators, *crossover* and *mutation*. Crossover is where information from two parents is combined to produce a new offspring, which will possess genetic information from each parent; this is analogous to biological reproduction. By applying this operator to join segments of strong candidate solutions, it is more probable that an improved solution will be produced as a result. Different recombination approaches can be taken, for example k-point or uniform crossover.

After crossover, the resulting offspring can then be mutated to stimulate further genetic diversification [5]. This has the effect of helping the population to avoid premature convergence, which is where solutions become too similar to one another and no progress can therefore be made. In mutation, elements of the offspring are altered using various stochastic techniques; such techniques include single-gene and swap mutation. Both the crossover and mutation methods, along with their rate of application, are crucial design decisions and are chosen to match the chromosome representation of solutions, to ensure they have the appropriate results on the population.

Termination criteria is also dependent on the problem at hand and can range from until a specific solution is found to until a predefined number of generations has been reached [48], or even until a certain time limit has been exceeded [49].

2.2.2 Application to Music

Evolutionary computation and genetic algorithms are prevalent in the art and musical domains. In chapter six of *The Art of Artificial Evolution* [50], Ramirez et al. generate an expressive performance computational model, achieved by applying a genetic algorithm, with single-point mutation and crossover, to the symbolic portrayal of a jazz musician's expressive performance. This model can then be used to produce impromptu performances, with similar characteristics to a human saxophonist. Musical expressiveness was also attempted to be captured by Grachten et al. [51] using evolutionary optimisation, i.e., through applying a GA, in order to expand performance notation. Another musical domain exposed to evolutionary computation is that of music improvisation. Biles [52] developed a system called GenJam, which is a GA-based model of an improvising jazz musician. Interestingly, here, fitness was evaluated in real-time by a human; this design decision is one which will be discussed further in the critical analysis section.

This project, however, is concentrating specifically on the realm of algorithmic music composition. One of the first links between composition and GAs was conceived in 1991 by Horner and Goldberg [53], which actually references a neural network resource used earlier in the report ([36]) as inspiration. The focus of Horner and Goldberg's paper is music composition through *thematic bridging*, which involves constructing a final pattern over a defined time period, given an initial one. Binary tournament selection and 1-point crossover were utilised, but unfortunately the results were not particularly successful, with the paper emphasising that the operation set, encoding and fitness function are all crucial areas which must be carefully considered when implementing a GA. In another study, Tokui and Iba [54] combine GAs with Genetic Programming (GP) [55] for music composition. Individuals in the GA population represent short rhythmic patterns and GP individuals express how these patterns are organised. The fitness of the GA population is evaluated through an ANN trained with a human subjective function. This method amalgamation allows musical structures to be adequately found in the search space, leading to euphonious rhythms being generated. NEUROGEN, developed by Gibson and Byrne [56], also used an ANN to evaluate fitness, albeit without human influence, to produce four-part harmony compositions. Phon-Amnuaisuk and Wiggins [57] produced four-part harmonisations too, using a GA as well as a rule-based system. They assert that the calibre of chorale melodies produced is highly reliant on the overall implicit and explicit knowledge of the system.

2.3 Technologies

There are a multitude of machine learning libraries available for use. Two of the most popular are TensorFlow [58] and Theano [59]. Both are Python [60] based libraries, with TensorFlow additionally supporting C++. Keras [61] is another Python library, acting as a high-level neural network API which can run on top of either TensorFlow or Theano. Magenta [62] is yet another interesting machine learning library, powered by TensorFlow, specifically targeting the music and art realms.

There also exist many impressive Python libraries for evolutionary computation, including Pyevolve [63] and Distributed Evolutionary Algorithms in Python (DEAP) [64].

On a different note, secure and scalable cloud computing can be rented out through Amazon Web Services (AWS) [65], which can massively speed up the training of neural networks by providing substantial processing power.

[Project specification omitted from this version of the document]

References

[1] K. Mcleod, "Genres, Subgenres, Sub-Subgenres and More: Musical and Social Differentiation Within Electronic/Dance Music Communities," *Journal of Popular Music Studies*, vol. 13, pp. 59–75, 2001.

[2] A. M. Turing, "COMPUTING MACHINERY AND INTELLIGENCE," tech. rep., 1950.

[3] M. J. M. J. Butler, *Unlocking the groove : rhythm, meter, and musical design in electronic dance music*. Indiana University Press, 2006.

[4] A. Nayebi and M. Vitelli, "GRUV: Algorithmic Music Generation using Recurrent Neural Networks," tech. rep.

[5] G. Nierhaus, *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer, 2010.

[6] D. Eck and J. Schmidhuber, "A First Look at Music Composition using LSTM Recurrent Neural Networks," tech. rep.

[7] Hermann Hild, Johannes Feulner, and Wolfram Menzel, "HARMONET: A Neural Net for Harmonizing Chorales in the Style of J.S.Bach," tech. rep., 1992.

[8] A. Huang and R. Wu, "Deep Learning for Music," tech. rep.

[9] W. S. McCulloch and W. H. Pitts, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, vol. 7, pp. 115–133, 1943.

[10] I. Rechenberg, "Evolutionsstrategie Optimierung technischer Systeme nach Prinzipien der biologischen evolution," tech. rep., 1973.

[11] D. Bahdanau, K. Cho, and Y. Bengio, "NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE," tech. rep., 2015.

[12] B. D. Ripley, *Pattern recognition and Neural Networks*. Cambridge University Press, 2007.

[13] N. Ganesan, K. Venkatesh, M. Rama, and A. Malathi Palani, "Application of Neural Networks in Diagnosing Cancer Disease Using Demographic Data," *International Journal of Computer Applications*, 2010.

[14] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation," tech. rep.

[15] J. Schmidhuber, "Deep Learning in Neural Networks: An Overview," apr 2014.

[16] A. L. Samuel, "Some studies in machine learning using the game of Checkers," *IBM JOURNAL OF RESEARCH AND DEVELOPMENT*, pp. 71—-105, 1959.

[17] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT*, vol. 16, pp. 146–160, jun 1976.

[18] G. Luger and W. Stubblefield, *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison Wesley Longman, 3 ed.

[19] J. Martens and I. Sutskever, "Learning Recurrent Neural Networks with Hessian-Free Optimization," tech. rep., 2011.

[20] M. C. Mozer, "A Focused Backpropagation Algorithm for Temporal Pattern Recognition," tech. rep., 1989.

[21] M. T. Hagan, H. B. Demuth, M. Beale, and O. De Jesus, *Neural Network Design*. 2 ed., 2014.

[22] A. Zell, "Stuttgart Neural Network Simulator User Manual."

[23] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A Novel Connectionist System for Unconstrained Handwriting Recognition," tech. rep.

[24] H. H. Sak, A. Senior, and B. Google, "Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling," tech. rep.

[25] X. Li and X. Wu, "CONSTRUCTING LONG SHORT-TERM MEMORY BASED DEEP RECURRENT NEURAL NETWORKS FOR LARGE VOCABULARY SPEECH RECOGNITION," tech. rep., 2015.

[26] S. Hochreiter and J. J. Urgen Schmidhuber, "Long Short-Term Memory," Tech. Rep. 8, 1997.

[27] S. Hochreiter, Y. Bengio, P. Frasconi, J. J. Urgen Schmidhuber, and I. S. C. Kremer, "Gradient Flow in Recurrent Nets: the Diiculty of Learning Long-Term Dependencies," tech. rep.

[28] R. B. Dannenberg, B. Thom, and D. Watson, "A Machine Learning Approach to Musical Style Recognition," *ICMC*, 1997.

[29] S. E. Fahlman and C. Lebiere, "The Cascade-Correlation Learning Architecture," tech. rep., 1991.

[30] F. J. Kiernan, "Score-based style recognition using artificial neural networks," *Proceedings of the first International Conference on Music Information Retrieval*, 2000.

[31] D. D. Johnson, "Generating Polyphonic Music Using Tied Parallel Networks," tech. rep.

[32] M. Matsugu, K. Mori, Y. Mitari, and Y. Kaneda, "Subject independent facial expression recognition with robust face detection using a convolutional neural network," 2003.

[33] "Oxford Learner's Dictionaries." `https://www.oxfordlearnersdictionaries.com/`, accessed 2018-11-21.

[34] M. C. Mozer, "Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multiscale processing," tech. rep., 1994.

[35] D. E. Rumelhart and J. L. McClelland, "Parallel Distributed Processing: Explorations in the Microstructure of Cognition," tech. rep., 1986.

[36] P. M. Todd, "A Connectionist Approach to Algorithmic Composition," Tech. Rep. 4, 1989.

[37] A. Swift, "A brief introduction to MIDI," *SURPRISE*, 1997.

[38] C. Darwin, *On the Origin of Species*. 1859.

[39] J. H. Holland, "Adaptation in Natural and Artificial Systems," 1975.

[40] D. Zwickl, "GENETIC ALGORITHM APPROACHES FOR THE PHYLOGENETIC ANALYSIS OF LARGE BIOLOGICAL SEQUENCE DATASETS UNDER THE MAXIMUM LIKELIHOOD CRITERION," tech. rep., 2006.

[41] D. Weile and E. Michielssen, "Genetic algorithm optimization applied to electromagnetics: a review," *IEEE Transactions on Antennas and Propagation*, vol. 45, pp. 343–353, mar 1997.

[42] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley, 1989.

[43] J. M. Weiss, "Genetic Algorithms and Sudoku," tech. rep., 2009.

[44] K. Sims, "Interactive Evolution of Equations for Procedural Models," *The Visual Computer*, vol. 9, no. 8, pp. 466–476, 1993.

[45] M. Haggerty, "Evolution by Esthetics, an interview with William Latham and Stephen Todd," *IEEE Computer Graphics and Applications*, vol. 11, pp. 5–9, 1991.

[46] T. Blickle and L. Thiele, "A Comparison of Selection Schemes used in Genetic Algorithm (2nd Edition). TIK Report No. 11.," vol. 2, no. 11, 1995.

[47] X. Yao, "An Empirical Study of Genetic Operators in Genetic Algorithms," tech. rep.

[48] D. Bhandari, C. A. Murthy, and S. K. Pal, "Variance as a Stopping Criterion for Genetic Algorithms with Elitist Model," *Fundamenta Informaticae*, vol. 120, pp. 145–164, 2012.

[49] S. N. Ghoreishi, A. Clausen, and B. N. Joergensen, "Termination Criteria in Evolutionary Algorithms: A Survey," 2017.

[50] J. Romero and P. Machado, *The Art of Artificial Evolution*. Springer, 2007.

[51] M. Grachten, J. L. Arcos, and R. López De Mántaras, "Evolutionary Optimization of Music Performance Annotation," tech. rep.

[52] J. A. Biles, "GenJam: A Genetic Algorithm for Generating Jazz Solos," tech. rep.

[53] A. Horner, "Genetic Algorithms and Computer-Assisted Music Composition," tech. rep., 1991.

[54] N. Tokui and H. Iba, "Music Composition with Interactive Evolutionary Computation," tech. rep., 2000.

[55] W. Banzhaf, P. Nordin, R. E. Keller, and F. D. Francone, "Genetic Programming An Introduction On the Automatic Evolution of Computer Programs and Its Applications," tech. rep., 1998.

[56] P. M. Gibson and J. A. Byrne, "NEUROGEN, musical composition using genetic algorithms and cooperating neural networks," *1991 Second International Conference on Artificial Neural Networks*, p. 383, 1991.

[57] S. Phon-Amnuaisuk and G. A. Wiggins, "The Four-Part Harmonisation Problem: A comparison between Genetic Algorithms and a Rule-Based System," tech. rep., 1999.

[58] "TensorFlow." `https://www.tensorflow.org/`, accessed 2018-11-21.

[59] "Theano." `http://deeplearning.net/software/theano/`, accessed 2018-11-21.

[60] "Python." `https://www.python.org/`, accessed 2018-11-21.

[61] "Keras." `https://keras.io/`, accessed 2018-11-21.

[62] "Magenta." `https://magenta.tensorflow.org/`, accessed 2018-11-21.

[63] "Pyevolve." `http://pyevolve.sourceforge.net/`, accessed 2018-11-21.

[64] "DEAP." `https://github.com/deap/deap`, accessed 2018-11-21.

[65] "Amazon Web Services." `https://aws.amazon.com/deep-learning/`, accessed 2018-11-21.