Application of Machine Learning Tools for Music Composition

COMP4560 Advanced Computing Project
Supervised by: Nick Birbilis
The Australian National University
Idris Mustafa
u6733671
Acknowledgment

The completion of this project could not have been possible without the assistance of so many people whose names may not all be enumerated. I would like to express my sincere gratitude to Professor Nick Birbilis for his support and guidance. I would also like to extend my gratitude to The Australian National University and the College of Engineering and Computer Science for providing this great Advanced Computing Project program.

To all the relatives, friends, and others who in one way or another shared their support, either morally or financially, thank you.
Abstract

Music composition using machine learning has developed a lot in the last few years. Music is an art that brings out the emotions, expressions, and creativity of an artist. Particularly, songs composed in the 1970s have a unique style and were totally dependent on the skills of an artist. Machine learning algorithms are able to capture the structure of music with ease. In this report, we explore various machine learning algorithms that are able to generate music similar to the songs composed by famous American guitarist, Nile Rodgers. Three different models have been used to generate music. First, we attempt to generate music using the first-order Markov chain, as a music piece is seen as a sequence of states with each state being the chords of the guitar. The results from Markov chain are random as it is conditioned only on one previous event. Second, we compose music using the Long Short Term Memory network that is designed to work efficiently for long temporal sequences. The results from LSTM are better sounding than the random sounds generated by Markov Chain. Third, a 1D Convolutional Neural Network is implemented. The training time improves drastically from the LSTM.

Keywords: markov chain, LSTM, music composition, convolutional neural network,
Contents

Acknowledgment 1

Abstract 2

1. Introduction 4
   1.1 Motivation 4
   1.2 Objective 5
   1.3 Related Work 5

2. Background 6
   2.1 Fundamentals of Music 6
       2.1.1 Musical Instrument Digital Interface 6
   2.2 First-order Markov Chain 7
       2.2.1 Transition Matrix 7
   2.3 Artificial Neural Network 8
       2.3.1 Recurrent Neural Network 9
       2.3.2 Long Short Term Memory Network 9
       2.3.3 Convolutional Neural Network 11

3. Experimental Methodology 15
   3.1 Reference Declaration 15
   3.2 Hardware 15
   3.3 Dataset and Preprocessing 15
   3.4 First-order Markov Chain experiment 16
   3.5 Deep learning experiment 16
       3.5.1 Long Short Term Memory Network settings 17
       3.5.2 1DCNN settings 17

4. Results 18
   4.1 Deep learning results 18
   4.2 Survey report 19

5. Discussion and Conclusions 21
   5.1 Future Work 21

Reference 22

Appendix: Independent Study Contract 24
1. Introduction

1.1 Motivation

Algorithmic music is interpreted as a collection of rules or procedures followed to produce a piece of music. The method and analysis of the algorithmic composition of music have been around for decades. These tools are being used by various music composers and performers as a helping hand. The software-generated music or art is often also sold by the companies or individuals that created it. An example of this is AIVA, a music generator which has released its own copyrighted albums.

The method of music generation conceptualized from the idea of Mozart’s Musical Dice Game proposed in 1787. Wolfgang Amadeus Mozart, a famous Roman musician, used 2 dice to randomly generate music from precomposed options. He manually created 272 tones from which he selected a tone based on the sum of the 2 dice[1]. Music generation has revolutionized since then. Music generation in the 20th century improved from randomly selecting sounds to applying concepts of maths to generate music algorithmically. French-Greek composer Iannis Xenakis, in the 1950s, used the concept of statistics and probability to generate music. He believed that the notes occurring in music is a sequence of elements that occur by chance. The selection of sounds was strictly based on probabilities and it was known as Stochastic Music[1]. His idea was closely related to the Markov process which became popular as many events in day to day life satisfies the Markov property. Rule-based Music generation by a computer was successfully experimented by Lejaren Hiller and Leonard Isaacson at the University of Illinois in 1956[1]. From there on, many researchers became more and more interested in applying various technological advancements to compose music. Artificial Intelligence has been successfully applied across many fields which have made it possible for machines to think like humans. AI methodologies are also applied in the field of music composition and have been successful in composing music. WaveNet[2] is one such model that is able to generate state of the art text to speech audio and is capable of composing good music.
1.2 Objective

In this report, we use three different models to generate music. Starting from a primitive model that is able to define music in terms of probabilities, we generate stochastic music using the first-order Markov chains. Markov chains are relatively easily applied to music compared to advanced deep learning algorithms, discussed in detail in section 3.4. Second, we experiment with a variant of Neural Networks, the Long Short Term Memory network which is discussed in section 2.3.2. Lastly, inspired by the WaveNet architecture[], we generate music using a 1D Convolutional Neural Network, discussed in detail in section. The aim of this report is to experiment with models that have proven to give good results as a generative model. With the models mentioned above, our goal is to generate music pieces that are similar sounding to the songs composed by Nile Rodgers but on its own is a unique variation.

1.3 Related Work

2. Background

In this section, we introduce some background knowledge required to understand the report. In section 2.1, the structure of music and the use of the MIDI file is discussed. In section 2.2, Markov chains are discussed in detail. Long short term memory network and Convolutional neural networks are explained in detail in the subsequent section 2.3.

2.1 Fundamentals of Music

Music is a form of art. The definition of music has changed over time, as communities of various cultures interpret music differently. Definitions vary from a practical and theoretical perspective to a philosophical perspective. For instance, the Greeks defined music as tones ordered horizontally as melodies and vertically as harmonies and according to philosopher Jacques Attali music is sonorous and can be determined as a sequence of events varying as sound and silence[7]. The core elements of music composed using instruments are pitch which is the frequency of a note's vibration, harmony which is the simultaneous combination of notes, also called chords, amplitude which is the loudness or softness of a note and duration which is the amount of time a note is played for.

2.1.1 Musical Instrument Digital Interface (MIDI)

Musical information is encoded in a midi file (Musical Instrument Digital Interface). A single midi link is capable of carrying up to 16 channels of information, each channel representing a separate device or instrument. Midi carries information in terms of event messages which specify music with some instructions relating to a note's notation, pitch, velocity, vibrato, etc.
2.2 First-order Markov Chain

Markov chain is a stochastic model where a system experiencing a transition from one state to the other can be modeled according to certain probability rules. Markov chain is “memory-less”. That is, the future states are not dependent on the actions that led to the current state. This is called the Markov property. There are many everyday processes that satisfy Markov property. A good example of the Markov property is a game of snakes and ladders or any other games whose actions are defined by the dice. The future state is decided based on the current position on the board and the next roll of dice and it does not depend on the actions that led to the current position.

In terms of random variables and conditional probability, a Markov chain is a sequence of $X_0$, $X_1$, $X_2$...$X_n$ random variables that are conditionally independent. The possible states of the random variables are $i_0$, $i_1$, $i_2$...$i_n$. The Markov property is given as,

$$P(X_n = i_n \mid X_{n-1} = i_{n-1}) = P(X_n = i_n \mid X_0 = i_0, X_1 = i_1, ..., X_{n-1} = i_{n-1}) \quad (2.1)$$

In a first-order system, the probability of the current event to occur is only dependent on the previous one event. Hence, the terms from 0 to n-2 in equation 2.1 can be ignored.

2.2.1 Transition Matrix

The probability of an event to occur conditioned on the previous event is given in a transition matrix. The rows and columns of a transition matrix $P$ at time $t$ for a Markov chain $\{X\}$ where {} represents the sequence of events is given as,

$$(P_{ij})_{ij} = P(X_{t+1} = j \mid X_t = i) \quad (2.2)$$

Transition matrix for an example sequence, Sequence - A,B,A,C,B,A,D,B,B,C,A

<table>
<thead>
<tr>
<th>Current Event</th>
<th>Future Event</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1/4</td>
<td>1/4</td>
<td>1/4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>2/4</td>
<td>1/4</td>
<td>1/4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>1/2</td>
<td>1/2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>1/2</td>
<td></td>
<td>1/2</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Table 1: Transition matrix*
2.3 Artificial Neural Network

One of the major tools used in Machine Learning techniques is the artificial neural network. They are intended to replicate a learning system similar to that of humans. ANNs consist of an input layer, hidden layer and an output layer. The hidden layer transforms the data from the input layer into a piece of meaningful information for the output layer. Each layer consists of components called neurons. Each neuron in the input layer is connected to every neuron in the hidden layer and each neuron in the hidden layer is connected to every neuron in the output layer.

Each connection is assigned a weight which implies the relative importance of a particular neuron. The input to the hidden layer neuron is computed as a weighted sum of all the connections from the input neurons connecting to the particular hidden layer neuron. A bias term is added to the result.

\[ Z = \sum_{i=1}^{n} x_i w_i + w_b \]  \hfill (2.3)

The output of hidden neurons is obtained by an activation function, for instance, sigmoid function, applied to the neurons.

\[ h_i = \sigma(Z) = \frac{1}{1+e^{-Z}} \]  \hfill (2.4)

2.3.1 Recurrent Neural Network
Recurrent neural network is a class of artificial neural networks which is applied on sequential information. Traditional neural networks assume that the input provided is independent of each other. In a Recurrent Neural Network, the output of a hidden layer at time step t is dependent on the output from time step t-1. The drawback of recurrent neural networks is it cannot remember information for longer sequences.

In figure 2, $X_0$, $X_1$, $...X_t$ imply the inputs at each time step. At time step 0, the network takes input $X_0$ and the network processes the input to generate a hidden state value $h_0$. At time step 1, the network takes the hidden state value from the previous node and an additional input $X_1$ which are processed together to output a hidden state value of $h_1$.

$$h_t = \sigma(W_o[h_{t-1},X_t]) \quad (2.5)$$

**2.3.2 Long Short Term Memory Network**

Long Short Term Memory Networks is a variant of Recurrent Neural Network that is able to learn the long term dependency in the sequence. They were first introduced by Hochreiter and Schidhuber (1997)[9].

All recurrent neural networks have a chain of repeating modules of neural networks with a simple structure, for instance, a single tanh layer. In standard RNN, the hidden state value of the previous module is combined with the input of the current processing module and the concatenated vector is passed through a tanh activation layer which produces a hidden state value for the current module. For longer sequences, the information has to travel through all the modules to reach the current processing module. In the learning process by backpropagation, the gradient value shrinks while updating weights from backpropagation...
through time and hence learning does not proceed effectively. This is called the vanishing gradient problem. LSTM architecture fixes this issue with the help of cell states. LSTMs also have repeating modules like the RNNs, but with a complex structure that helps them remember information for a longer period.

In Figure 3 (Chris Olah 2015), the line going from one module to another carries a vector from the output of one module to the input of the next module. The lines that merge into one denote concatenation operation and the lines that split denote that the information is being copied. The horizontal line on the top indicates the cell state and information can be added or removed to the cell states with the help of gates. The first step is to decide which information needs to be discarded. This decision is made by the forget gate which uses the sigmoid function. It looks at $h_{t-1}$ and $x_t$ and outputs a number between 0 and 1. None of the information is passed for a value of 0 and all the information is passed for a value of 1.

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$$

In equation, $f_t$ refers to the information to be discarded.

The second step is to check which information should be stored in the cell state. This is done with the help of a sigmoid and a tanh layer. The sigmoid layer decides which information to update and the tanh layer creates a new vector to be added to the cell state. Tanh layer is used to overcome the vanishing gradient problem by sustaining the second
derivative over a long range before reaching 0. The output from sigmoid and tanh is combined to create an update to the state.

\[ i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \ast \tanh(W_C[h_{t-1}, x_t] + b_C) \] (2.7)

The old cell state is updated to the new cell state \( C_t \) after having computed the information to be discarded and the information that is to be inputted to the cell state.

\[ C_t = f_t \ast C_{t-1} + i_t \] (2.8)

\( f_t \) in equation 2.8 indicates the information that has been discarded at time step \( t \), \( C_{t-1} \) indicates the old cell state, \( i_t \) is the new selected input, obtained from equation 2.7, that should be in the cell state. The final step is to decide what information needs to be output. This is done with the help of a sigmoid layer and a tanh layer. The sigmoid layer selects which information of the cell state should it output. Also, the current cell state is passed through a tanh layer which scales the value in the range -1 to 1. The output of the tanh layer is multiplied with the output of the sigmoid layer to only output the selected information. Hence, our new hidden state is,

\[ h_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \ast \tanh(C_t) \] (2.9)

The hidden state is passed further to the next module which also processes a new input of time step \( t+1 \).

### 2.3.3 Convolutional Neural Network

Convolutional Neural Network, also known as ConvNet, is a class of deep neural networks which is popularly used for analyzing visual imagery. Its success in computer vision tasks has motivated researchers and practitioners to apply as a generative model. Convolutional Neural Network is similar to ANN except for, the input to the neurons is transformed into more meaningful information using convolutional layers. A convolution operation is performed on the input using these convolutional layers and the result of convolution is passed to the next layer. The key idea is similar to the response from neurons in the visual cortex of humans to a specific stimulus. This is called the receptive field. The receptive field in the convolutional neural network is the particular region in the input of any layer that affects a part of the network. The convolutional layer looks at only a subarea of the input. This subarea is usually a square region (for instance, 3 x 3 kernel) for a two-dimensional input such as an image. In a fully connected layer, the receptive field is the entire input from the previous layer.
A convolutional neural network has different layers of neurons. The input is passed to the convolutional layers, the output from which is passed to a fully connected layer of neurons. The fully connected layer then outputs the desired value for the task. Each convolutional layer may be passed through a pooling layer where the dimension of the output from convolution is reduced. As our focus is to work on music data, which is a one-dimensional sequence, the convolution operation discussed further is only for one-dimensional convolutional layers.

Each convolution layer has the following attributes,

- A convolutional kernel, usually defined by its width and height.
- Type of padding used.
- Strides.

The convolution operation is done by multiplying the weights specified in the kernel with the input subarea and summing over all the products. To obtain the output of a convolutional layer, the kernel is convolved over a particular region of the input. An illustration of convolution operation with a kernel size of 3 is shown in Figure 5. The output \( c_3 \) is given as,

\[
c_3 = w_1i_2 + w_2i_3 + w_3i_4
\] (2.10)
The output of convolution also depends on the type of padding used. Padding is the amount of extra nodes added to the input sequence. For instance, a padding of zero would add a new value of zero before the first index of the input and after the last index of the input. Hence, $c_1$ and $c_6$ in Figure 5 can be calculated as,

\[ c_1 = 0 + w_2i_1 + w_3i_2 \]
\[ c_6 = w_1i_5 + w_2i_6 + 0 \]

There are various types of padding. For instance, padding used in our model is a causal padding, generally used for temporal sequences, that pads zeros only before the start of the input sequence. This is done to help predict values of earlier time steps.

Stride is the amount of shifting of the kernel over the input sequence. For instance, a stride of 1 would mean that the kernel is shifted by 1 index at a time and a stride of 2 would mean that the kernel is shifted by 2 index at a time.
The convolution output is then processed through a pooling layer whose job is to reduce the complexity of the features generated. A commonly used pooling method is the max pooling.

![Figure 7: Illustration of max pooling](image)

In figure 7, max pooling is applied to the sequence using a pooling size of 2 and stride of 2. The pooling kernel looks at a subarea in the feature space starting from the first index and outputs the maximum value found in that subarea of the feature space.
3. Experimental Methodology

In this section, we introduce in detail the experimental settings used and the procedure followed.

3.1 Reference Declaration

A playlist of 26 songs from the artist, Nile Rodgers was provided by Professor Nick Birbilis to use for the experiment. As the music files were not in MIDI file format, Chordify.net was used to extract the guitar chords progression for all the songs to prepare the input for our models.

3.2 Hardware

All the experiments were developed and run on a laptop with a 6th gen Intel Core i5 processor. Limited by the hardware, the experiment settings used are focussed to produce music pieces with reasonable training time.

3.3 Dataset and Preprocessing

The songs provided contained various other instruments which were not necessary for our experiment. As these songs are the original work published by the artist, the MIDI files were not easily available on public repositories. Directly converting the mp3 files to MIDI resulted in various instruments being played in a single MIDI channel. Hence, the MIDI files for the specified songs were manually picked from chordify.net. However, the files also contained certain piano notes present in the song. The piano notes were in another channel and could be separated easily.
Chords are formed by the simultaneous combination of 2 or more notes of different pitch. Hence, they are combined to form a single vector. The data is further simplified by assigning an integer value for every unique chord formed.

### 3.4 First-order Markov Chain experiment

Markov chains do not have learning similar to other machine learning algorithms. It does not learn from the data. Rather, it uses the whole data to generate the transition matrix which is used to directly predict the output.

```
Algorithm 1 To compute transition matrix for a first-order Markov chain.

Input: A sequence of chords transformed to integers \{0, 1, \ldots, n\}
Output: Transition matrix \(P\) of dimension \(R^{n \times n}\)

1: for all \(i = 0, 1, \ldots, n\) do
2:  for all \(j = 0, 1, \ldots, n\) do
3:    set: \(P_{i,j} = P(j_n | i_{n-1})\)
4:    set: \(P_{j,i} = P(i_n | j_{n-1})\)
5:  end for
6: end for
```

The transition matrix is generated using Algorithm 1. To generate music, a random chord is first selected as a starting point. The chord after the first one is sampled based on the probabilities from the transition matrix. Further chords are generated by incrementing the pointer to set the current event as the newly sampled chord.

### 3.5 Deep learning experiment

In this subsection, the settings used for the Long Short Term Memory network and 1D convolutional neural network are discussed. The key idea to generate music using both of the models is to feed the network with input sequences and predict chords from a list of chords found in the input space. The inputs to the model are prepared by taking a sequence of chords up to time step \(t_n\) and their corresponding target as the chord occurring at time step \(t_{n+1}\). In other words, the chord at time step \(t_{n+1}\) is predicted based on the \(n\) previously occurring elements. Hence, \(n\) can be said as a lookback parameter for our model. Lookback
parameter in the case of music composition gives the sequence varying length phrasal structures. This means that for a higher value of lookback parameter, a longer phrase from music is being used to learn and generate new chords. We experiment with different values of the lookback parameter as an attempt to get a piece of better sounding music from a considerably small dataset of ours.

The neurons in the output layer for both the deep learning models are the unique chords found in the input space we try to predict. There are 54 unique chords in our dataset.

### 3.5.1 Long Short Term Memory Network settings

The LSTM network used is a three layer network, first two layers are the LSTM layers consisting of 128 neurons, directly connected to the output layer of 54 neurons. Tanh activation function is used for the two LSTM layers. The output layer is activated using the softmax function which outputs the neurons as a probability distribution. The model is trained for 50 epochs. The new chord is determined as the chord with the maximum probability in the predictions.

### 3.5.2 1D Convolutional Neural Network settings

The 1D CNN used consists of 5 layers of convolution each activated using the relu activation function as it is computationally fast compared to other activation functions. The output from the final convolutional layer is connected directly to the output layer of 54 neurons activated by the softmax function.
4. Result

In this section, the results from two deep learning models are compared. Results from Markov chain are not compared with the LSTM and 1DCNN, as discussed earlier in section 3.4, the learning is different from the deep learning models. The music piece generated from a Markov chain is only compared subjectively through a survey report. The survey report also includes the comparison of music pieces generated from LSTM and 1DCNN.

4.1 Deep learning results

For different values of the lookback parameter, the learning pattern of both the models is almost the same. However, models with different values cannot be compared in terms of training loss or accuracy, as the target values also change when preparing the data for training with different lookback values.

![Figure 8: Training Loss, Left - Lookback set to 20, Right - Lookback set to 30](image)

Training loss of 1DCNN in both the settings shown in figure 8 decreases initially and slowly starts to flatten out in the end. Test set accuracy, shown in figure 9, does not improve for 1DCNN in both the settings.
4.2 Survey report

The music pieces generated from our models were subjectively evaluated through a survey report. The survey report consisted of six audio samples, three of which were the outputs from the three models, and the other three were parts of the input. Individuals were asked to identify the audio samples as either human-composed or computer generated.

The music piece generated using a markov chain obtained 45.8% votes as human-composed. The piece that obtained the least number of votes as human-composed at 25%, was from the 1DCNN model. The piece generated by the LSTM model got the maximum votes as human-composed at 54.2%.
Figure 11: Music piece generated from the LSTM network

Figure 12: Music piece generated from the 1DCNN
5. Discussion and Conclusions

We have generated music pieces from three different models. The survey report gives a general idea of how the pieces generated from the models are being misclassified as human-composed. A proper conclusion cannot be drawn from only the survey report as an individual's subjective opinion may be different from others. However, from the generalization accuracy results shown in figure 9, we can conclude that the LSTM network is able to learn better on our dataset compared to the 1DCNN. Convolutional networks give poor generalisation performance and could perform better if it were provided with more training samples. LSTM performs best on our dataset due to its gated architecture to remember information from sequences whereas 1DCNN does not store information about hidden states from previous time steps. They only generate new features with the help of convolution filters. Hence, more training data would be required in order to efficiently make use of the convolutional layers to improve the generalisation performance.

5.1 Future Work

In this paper, we have attempted to predict guitar chords progressions using machine learning tools. The approach taken for encoding the chords occurring sequentially in the midi file does not take into consideration the duration of each chord being played. The project can be extended by effectively encoding the duration of each chord together to compose better music. Also, as the dataset is of limited size, transfer learning approach can be a possible next step for our work where pretrained models trained on a larger dataset could be fine tuned on our dataset.
Reference


[12] Cs231n.github.io. Available: https://cs231n.github.io/assets/cnn/stride.jpeg. Figure 6 on page 13

[13] [Online]. Available: https://www.researchgate.net/figure/Calculations-involved-in-a-1D-convolution-operation_fig3_324177888. Figure 5 on page 13

[14] [Online]. Available: https://www.researchgate.net/figure/1D-max-pooling-operation_fig4_324177888. Figure 7 on page 14
List of Figures

Figure 1 - Artificial Neural Network ......................................................... 8
Figure 2 - Recurrent Neural Network ......................................................... 9
Figure 3 - Long Short Term Memory Network .......................................... 10
Figure 4 - Convolutional Neural Network .................................................. 12
Figure 5 - Convolution operation ............................................................... 13
Figure 6 - Illustration of stride ................................................................. 13
Figure 7 - Max Pooling ........................................................................... 14
Figure 8 - Training Loss ......................................................................... 18
Figure 9 - Test set accuracy ................................................................. 19
Figure 10 - Music survey for first-order Markov Chain ...................... 19
Figure 11 - Music survey for Long Short Term Memory Network .......... 20
Figure 12 - Music survey for 1DCNN ..................................................... 20
Appendix: Independent Study Contract

INDEPENDENT STUDY CONTRACT
PROJECTS

Note: Enrolment is subject to approval by the course convenor

SECTION A (Students and Supervisors)

UniID: __ut6733671________________

SURNAME: Mustafa Chatriwala FIRST NAMES: Idris

PROJECT SUPERVISOR (may be external): Professor Nick Birbilis

FORMAL SUPERVISOR (if different, must be an RSECS academic): Professor Nick Birbilis

COURSE CODE, TITLE AND UNITS: COMP4560 Advanced Computing Project - 12 units

COMMENCING SEMESTER: S1 ✗ S2 YEARS: 2019 Two-semester project (12u courses only): ✗

PROJECT TITLE: Application of Machine Learning tools for Music Composition

LEARNING OBJECTIVES:
To utilise machine learning as means for generating original content (e.g., training).
To learn and deploy machine learning as a tool in music generation.

PROJECT DESCRIPTION:

Google's open source project Mjent0, a machine learning tool, will be utilised and explored for its ability to potentially generate new music based on the data on which a machine learning model is trained.
The project focuses on music generation (not lyrics) and will explore music from a particular genre as the training input.
The exploration of Mjent0, which is rather new, has not been done in broad detail previously and the project will make original investigations and contributions.

Research School of Computer Science

Form updated Jan 2018
ASSESSMENT (as per the project course’s rules web page, with any differences noted below).

<table>
<thead>
<tr>
<th>Assessed project components:</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Report: style:</td>
<td></td>
<td>(min 45, def 60)</td>
<td>(examiner)</td>
</tr>
<tr>
<td>(e.g. research report, software description...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Artefact: kind:</td>
<td>(max 45, def 30)</td>
<td>(supervisor)</td>
<td></td>
</tr>
<tr>
<td>(e.g. software, user interface, robot...)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Presentation</td>
<td>(10)</td>
<td></td>
<td>(course convener)</td>
</tr>
</tbody>
</table>

MEETING DATES (IF KNOWN):

STUDENT DECLARATION: I agree to fulfil the above defined contract:

.................................................. 28-7-2019............................
Signature                     Date

SECTION B (Supervisor):

I am willing to supervise and support this project. I have checked the student’s academic record and believe this student can complete the project. I nominate the following examiner, and have obtained their consent to review the report (via signature below or attached email)

.................................................. 26-7-2019............................
Signature                     Date

Examiner:

Name: Ben Swift........        Signature
(Nominated examiners may be subject to change on request by the supervisor or course convener)

Research School of Computer Science  29/07/2019 Form updated Jun 2018