On the Evaluation of Music Generation Algorithms Using Music Theory

Jan-Felix De Man
Department of Data Science and Knowledge Engineering
Maastricht University
Maastricht, The Netherlands

***Abstract*— Music generation is a rapidly evolving trend in machine learning. Problems that arise when generating something as subjective as music, is a proper way of evaluation. This work discusses a method of evaluation and compares four music generating models according to this method. The models used in the work are the BasicRNN, the Music Transformer, the MusicVAE and MuseGAN. The evaluation method, consisting out of 9 features related to music theory, will be used to compare these models against each other if a fair comparison can be made. The evaluation feature that proved to be the most useful is the pitch class transition matrix. This work will discuss how the BasicRNN outperforms the Transformer in terms of mimicking a song, but how the Transformer sounds more interesting. The different sampling modes in MuseGAN will be analysed and that a higher temperature when sampling from the MusicVAE usually leads to a more sophisticated samples.**

I. Introduction

Music has always been something defined and created by humans. Many animals can't even perceive music like we do. It is impossible to state when music got invented, because we can only assume that the first human beings also had this feeling for rhythmic percussion and singing that we all have in us, to a certain extent. While humanity has claimed music to themselves, some among us managed to create algorithmic compositions. It was as early as 1957 that Mathews, from Bell Laboratories, managed to generate the first musical piece. The advanced rule-based system he implemented, combined with markov chains or L-systems, led to the composition that was named the 'Illiac Suite'. By using a Monte Carlo algorithm for generating random sequences and a selection procedure based on the theoretical assumptions of Information Theory (which states that how improbable a message is to occur at a certain point, the more informative it is), it managed to generate compositions from scratch [7].

This method could still be considered very human. The rule-based system was based on music theory created by humans and influenced by the preference of the programmer. This resulted in very natural compositions, that followed classical music theory and did indeed sound like they were made by a person.

More recently, music generation has been imported to neural networks. The increase in available data and computing power, allows one to capture music in a more complex model, that is able to create its own rules and preferences. Many machine learning approaches towards music generation have been implemented over the past decade, often inspired by model architecture that is used for natural language processing [3].

As music is a very subjective thing, it is hard to compare and evaluate any kind of music. This work will discuss some limitations towards music evaluation as proposed by Yang and Larch [14], how state-of-the-art machine learning models compare based on this evaluation, what the limits are of these models and how they differ from each other. Most of the models that will serve for comparison are provided by an open-source Google project called Magenta. The models, described in Subsection IV-A, are the BasicRNN, the MusicVAE and a Transformer. A fourth model, called MuseGAN, a GAN model that can generate multi-track sequences, will also be discussed. MuseGAN has two different options to generate music from scratch: inference and interpolation. While the first uses random noise drawn from the latent distribution, the second draws the noise as grid. The MusicVAE and the BasicRNN can influence their output by a scalar called temperature, which allows the model to differ more, or less, from input. The following research questions will be answered:

- What differentiates the samples from interpolation and those from inference in the MuseGAN model?
- Does the Transformer outperform the BasicRNN when continuing on a primer sequence?
- What is the effect of the temperature of the sampling mode of the MusicVAE when generating from scratch?

II. Background

Music is defined by the way it sounds. Humans have developed an efficient way to spread music way before there was any means of recording it: sheet music. Every note one can play on the piano, one can pen down as sheet music. Figure 1 is an example of sheet music. It shows which notes

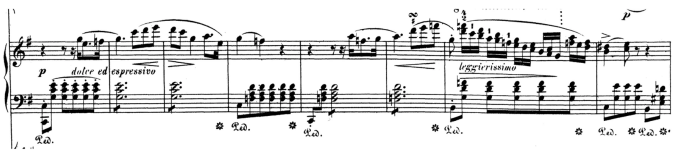

Fig. 1: Excerpt from the score of Chopin's Piano Concerto No. 1, taken from [8]

to play, when to play them and their duration. It has its

notation for silence, called rests. It contains other instructions on the dynamics of the song, such as volume changes: crescendos and decrescendos, from loud to quiet and vice versa respectively, and instructions on the modality, such as leggierissimo, as seen in Fig. 1, which means very lightly.
Most music generation models, however, do not work with audio files nor sheet music as input, but instead they use the Musical Instrument Digital Interface, or midi data. These, in turn, are often converted to numpy arrays or other encodings such as one-hot. Midi can be compared to a digital file

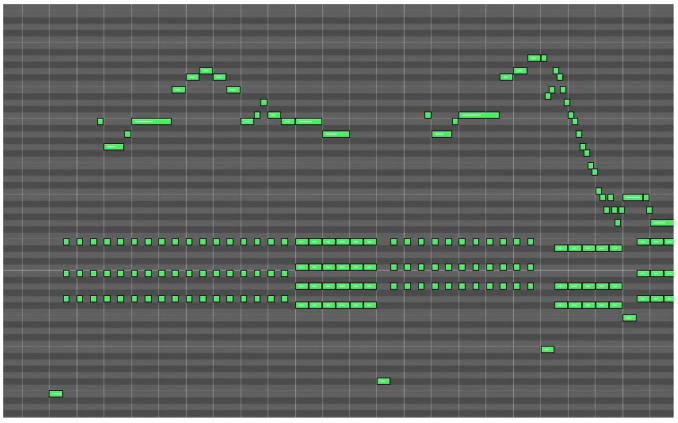

Fig. 2: Piano roll based on the score in 1. The horizontal axis represents time; the vertical axis represents pitch; each rectangle is a note; and the length of the rectangle corresponds to the duration of the note, taken from [8]

containing chord music for all the instruments of a song. Each of the tracks, or pianorolls, define what each instrument should play. As seen in Fig. 2, a pianoroll visualization of Fig. 1, one can easily map all the notes and rests from sheet music to a midi file. What gets lost in this conversion, however, are all the composer's suggestions on dynamics. A midi file can be converted to an audio format, what results in every note having the same volume and all of the notes being played at exactly the right time. This makes the music generated by algorithms often sound very inhumane and mechanic. The Illiac Suite was performed by a string quartet after the sheet music was generated. The sheet music did not include dynamics, but the interpretation of the musicians, and the fact that its working was still very influenced by humans, as mentioned in Section I, lead to a very natural result.

III. Related Work

Yang et Larch [14] proposed a method that can be applied to analyse characteristics or to have an objective evaluation with interpretable metrics for music. In search of formative evaluation for (generated) music, they developed a reproducible, reliable and comparable method for assessment. The implementation of absolute measures, of which the roots arise in musical domain knowledge, serve as a tool for analysis of quantifiable characteristics in the chosen set.

One of the models that will be analysed in this work is a Variational Autoencoder (VAE), the MusicVAE as presented by Roberts et al.[10]. It prides itself in the implementation of a novel hierarchical decoder. In contrast to other VAE's, of which the decoder usually exists out of a simply stacked RNN's, it passes the latent vector z through a 'conductor' RNN, which produces an embedding for every subsequence. All of these embeddings are passed through a fully-connected layer with $tanh$ activation, that produces initial states for a final decoder RNN. This decoder autoregressively outputs a sequence of distributions for each subsequence via a softmax output layer.
This novel architecture, illustrated in Fig. 20, allowed them to improve the standard VAE music generation models both quantitatively and qualitatively.

Another way of creating a latent space to draw samples from, is by using memory embeddings such as in a Transformer. The novel addition added by Huang et al. [6] to standard Transformer models, was a memory reduction of the relative attention embeddings. Since musical sequences tend to be bigger than for example language sequences, for which the Transformer was originally used (Parikh et al. [9]), a memory complexity quadratic to the sequence length is undesirable. The introduced skewing procedure achieves to reduce the memory complexity to linear, while also speeding up the computing time by factor 6 for a sequence of length 650.
And while the main focus in the field of music generation is on generating piano music. Dong et al.'s MuseGAN [4], following in the footsteps of MidiNet (Yang et al. [15]), proposes 3 GAN models for multi-track music generation. These models can generate up to 5 tracks, thus 5 instruments, in their samples. Their innovative temporal structure embedding, described in Subsection IV-A4, allows the models, that generate bar per bar, to remain a more coherent whole between all the separate bars.
This work will perform an analysis on the models mentioned above and compare the different methods available for sampling from them.

IV. Methods

A. Methods for generation

1) BasicRNN: The BasicRNN is the first demo model launched by Google's Magenta project. As the name suggests, it is a recurrent neural network (RNN), which means it has looped or recurrent connections, allowing the network to hold information accross inputs. For the BasicRNN, Long Short-Term Memory (LSTM) cells were used. The network is implemented implemented as a feed-forward network, that is trained using a gradient descent technique called backpropagation through time. The model implements a one-hot encoding to represent extracted melodies to the LSTM's. It is trained with MIDI files transposed to the pitch range $[48, 84]$ such that its output would also be in this interval. This corresponds to the 28^{th} and 64^{th} note on the piano. It is meant for generating

continuations to an input sequence, often called the primer sequence. This model is possibly the least interesting presented in this work, but will serve as a baseline for one of the experiments.

2) MusicVAE: Google Magenta's MusicVAE is a Variational Auto Encoder, a deep latent variable model. Similar to the regular Auto Encoders, it uses a encoder and decoder. The encoder is a series of convulational layers that compress the data into a lower dimension. This is called the latent space representation. The decoder, also existing out of a series of deconvulational layers, tries to reconstruct the original data from the latent space. During training, the loss, calculated as the difference between the original data and the reconstructed data, is minimised. The flattened latent space obviously has a lower dimension, while the input layer has the same dimension as the output layer. The latent space, or bottleneck, is the most important characteristic of auto encoders, as it compresses it will extract useful features and properties. What differentiates a Variational Auto Encoder (VAE) from a regular Auto Encoder, is that the latent space of a VAE consists out of latent distributions, instead of just values. This entails that from these distributions, one can sample the attributes. From these sampled attributes, the decoder can generate new output. Such models model both $p(z|x)$ and $p(z)$, where z is the latent vector that can either be inferred from existing data or sampled from a distribution over the latent space. A downside of (variational) auto encoders is that they work with continuous-valued data such as images. Modelling sequential data typically requires an autoregressive decoder, which is often sufficiently powerful for the autoencoder to ignore the latent space (Bowman et al. [2]). Roberts et al. [10] introduced a sequential autoencoder using a hierarchical recurrent decoder, the workings of which are illustrated in Fig. 20. By encoding an entire sequence to a single latent vector, it allows the model to capture longer sequences. The encoder, $q_\lambda(z|x)$ is a recurrent neural network (RNN). It processes the input sequence and produces a sequence of hidden states, in function of which the parameters of the distribution of the latent code z are set. The decoder, $p_\theta(x|z)$, sets the initial state of a decoder RNN using the sampled latent vector z. It autoregressively produces the output sequence. As with regular VAE's, the model is trained to recreate the input sequence and to train an appropriate encoder close to the prior latent space.[10]

3) Transformer: Like many other models, the music Transformer, another Google Magenta project, takes a language-modelling approach (Vaswani et al. [13]) and implements it for music generation. Transformers use a sequence-to-sequence architecture. Just like autoencoders, this makes them very useful for things such as translation, and both consist out of an encoder and a decoder. The transformer's decoder is an autoregressive generative model, it primarily uses learned or sinusoidal position information. Each of its layers consists out of a self-attention sub-layer followed by a feedforward sub-layer. The idea behind it is similar to an LSTM, but instead of an RNN architecture, it is a feedforward network. Fig. 21 portrays the main architecture of a Transformer. Attention layers first transform the L D-dimensional input vectors X into Q, K and V. $Q = XW^Q$ and contains the vector representation of one element in the sequence. $K = XW^K$ and represents all the elements in the sequence. $V = XW^V$ and contains the values of all the elements in the sequence. W^Q, W^V and W^K are all square matrices of size D. Each $L \times D$ query, key and value matrix is then split into H $L \times D_h$ attention heads, indexed by h. The attention heads have dimension $D_h = \frac{D}{H}$. A sequence of vector outputs is then computed for each head using the scaled dot product:

$$Z^h = Attention(Q^h, K^h, V^h) = Softmax(\frac{Q^h K^{h\top}}{\sqrt{D_h}})V^h \tag{1}$$

Using the dot product allows the attention heads to perceive the intersequential relation between the notes. Two notes from the same chord in the used key, will probably have a higher value in the embedding than two random notes. There are multiple attention heads, which allows the model to give different context to the same notes. A note following a chord progression at one point, is not necessarily as relevant later in the sequence when playing another chord, the multiple heads allow multiple outputs for the same notes. The outputs for each head are concatenated and linearly transformed such that Z becomes a $L \times D$ matrix. In the model implemented, the feedforward sub-layer then takes output Z from the previous attention sub-layer and performs two layers of point-wise dense layers on the depth D dimension.

$$FF(Z) = ReLU(ZW_1 + b_1)W_2 + b_2 \tag{2}$$

Shaw et al. [12] introduced relative position representations to allow attention to be informed by the distance in position between two elements in a sequence. A separate relative position embedding E^r of shape (H, L, D_h), that has an embedding for each possible pairwise $r = j_k - i_q$ between a query and key in i_q and j_k respectively. The embeddings are set separately for each head and are ordered by distance, which is a value between $-L + 1$ and 0. These embeddings interact with queries to compute S^{rel}, an $L \times L$ dimensional logits matrix. For each head the relative attention is computed as:

$$RelativeAttention = Softmax(\frac{QK^\top + S^{rel}}{\sqrt{D_h}})V \tag{3}$$

Shaw et al.'s implementation has a total space complexity of $O(L^2 D)$, but using a skewing procedure, Huang et al. [6] managed to reduce this to $O(LD + L^2)$. This memory requirement is still too high for very long sequences, therefore relative local attention is introduced, according to Huang et al., The concept is to divide the input sequence into blocks that do not overlap. Each block then attends only to itself and the block before.

4) MuseGan: MuseGan is short for multi-track sequential generative adverserial network (GAN). The core of a GAN, as introduced by Goodfellow [5], is to achieve adversial learning by constructing two networks: the generator and the

Model	BasicRNN	MusicVAE	Transformer	MuseGAN
Main architecture	RNN	CNN	CNN	CNN
Latent space		✓	✓	✓
Generate from scratch		✓	✓	✓
Continue on primer sequence	✓		✓	
Generate accompaniment			✓	✓
Temperature tweaking	✓	✓		
Interpolation		between sequences		sampling

TABLE I: Overview of main characteristics of the generative models

discriminator. The discriminator D is trained to distinguish the random noise z sampled by the generator G. They are in a feedback loop with each other, which allows the training procedure to be described as a two-player minimax game:

$$\min_G \max_D E_{x\sim p_d}[log(D(x))] + E_{z\sim p_z}[1 - log(D(G(z)))] \quad (4)$$

with p_d the distribution of the real data and p_z the prior distribution of z. The model used, uses Wasserstein distance instead of Jensen-Shannon divergence, as proposed by Arjovsky et al. [1]. This yields faster convergence to better optima and requires less parameter tuning, thus the objective function of D becomes:

$$E_{x\sim p_d}[D(x)] - E_{z\sim p_z}[D(G(z))] + E_{\hat{x}\sim p_{\hat{x}}}[(\nabla_{\hat{x}}||\hat{x}||-1)^2] \quad (5)$$

where $p_{\hat{x}}$ is defined by sampling uniformly among straight lines between pairs of points sampled from p_d and p_g, the model distribution.

The input $\bar{z}$ to MuseGAN consists out of 4 components: time independent random vectors z and z_i, which are inter-track and intra-track respectively. Time dependent random vectors z_t and $z_{i,t}$, again respectively inter-track and intra-track. On Fig. 22 the workings of the MuseGAN and these 4 components are displayed.

There is a shared temporal structure generator G_{temp} that takes the time-independent vectors z_t and a private temporal structure generator $G_{temp,i}$ for the time-dependent random vectors $z_{t,i}$ for all tracks i. This novel feature is implemented such that there would be coherence between different bars, as it models the temporal structure of a track. The generator thus consists out of two sub networks: the bar generator G_{bar} and the temporal structure generator G_{temp}. As mentioned above, G_{temp} takes in z_t and $z_{t,i}$ and maps it to some latent vectors $\overrightarrow{\mathrm{z}}$ which holds the temporal information. $\overrightarrow{\mathrm{z}}$ is then used by G_{bar} to generate music bar per bar. As the above method is for generating music from scratch, using the GAN to complete a song, another encoder needs to be added. With bar sequence $\overrightarrow{\mathrm{y}}$ and time-dependent random noise $\overrightarrow{\mathrm{z}^{(\mathrm{t})}}$ as input, an additional encoder, E, needs to be added. E is trained to map $\overrightarrow{\mathrm{y}^{(\mathrm{t})}}$ to the space of $\overrightarrow{\mathrm{z}^{(\mathrm{t})}}$. This way, the encoder is expected to extract the inter-track features of the given track instead of the intra-track features.

B. Comparison of the generative models

As can be deducted in table I, three of the models are able to generate from scratch. MusicVAE, Transformer and MuseGAN do this by drawing from their latent space. What differentiates the models in terms of input, is that the Transformer and BasicRNN can take in a short fragment, called a primer sequence, and predict a next sequence, thus 'completing' the song. The MusicVAE can not take in one fragment, but is able to interpolate two sequences, which results in a smooth transition from one song to another. The MuseGAN has two ways to generate samples, either by inference or interpolation of the latent space. Both the Transformer and the MuseGAN have a conditional model, which can be used to generate accompaniment for the input sequence.
The inner workings of these models are very different. The BasicRNN's aim is to mimic patterns seen in the training data and primer sequences. The MusicVAE generates samples by drawing elements out of the latent space distribution. The MuseGAN's generator creates the latent space by trying to fool the discriminator with sequences that it would classify as real. Sampling is then done from this latent space. The Transformer, much like the GAN and the VAE, draws from its latent space, but this latent space is created according to the attention mechanics.

C. Method for evaluation

Evaluating music is a very subjective thing. What sounds good to one person, might not sounds good to someone else. Although there exist many professional music critics and among them there, supposedly is a general consensus on what sounds good and what doesn't, using these resource can easily become very expensive and still not very objective. Yang and Lerch[14] proposed the evaluation of generated music based on 9 key features: 5 of which harmonic, 4 of them rhythmic measures. Their objective method provides a good baseline to distinguish between good and bad samples.

1) 5 features for harmonic evaluation: The 5 features as proposed by Yang and Lerch are the following:

- Pitch count (PC): a scalar that counts the amount of different pitches within a sample.
- Pitch class histogram (PCH): an octave-independent representation of the pitch content. It has dimensionality 12, which is the amount of pitches in an octave.
- Pitch class transition matrix (PCTM): A two-dimensional, histogram-like representation that is computed by counting the pitch transitions for each ordered pair of notes.
- Average pitch interval (PI): a scalar that is the subtraction of the highest and lowest used pitch within each sample.

2) 4 features for rhythmic evaluation:

- Note count (NC): a scalar that represents the amount of notes in a sample. It has no regard for pitch nor duration of the notes.
- Average inter-onset-interval (IOI): the average time between notes, or more simply, the average duration of silence in between notes. It is a scalar in seconds for each sample.
- Note length histogram (NLH): a histogram counting the occurences of the predefined allowable beat length classes [full, half, quarter, 8th, 16th, dot half, dot quarter, dot 8th,

dot 16th, half note triplet, quarter note triplet, 8th note triplet]. The values are quantized such that they would fall under one of the length classes.
- Note length transition matrix (NLTM): a two-dimensional representation of the rhythmic transition between notes following each other, similar to the pitch class transition matrix.

V. Experiments

Using the evaluation method introduced by Yang et Larch [14], the generated samples were evaluated and analyzed. Three experiments have been conducted:

- Comparison of MuseGAN samples from inference and interpolation.
- Comparison of PerformanceRNN and Transformer evaluated by the 9 key features
- Comparison of the effect of temperature using the MusicVAE and PerformanceRNN models

A. Experiment 1: inference and interpolation in a Generative Adversarial Network

Generative adverserial networks, such as MuseGAN, can generate output in two different ways: either inference or interpolation. Inference generates samples using random noise drawn from the latent distribution using training. Interpolation draws from the same latent space, but instead the noise inputs are drawn as a grid.
One would expect the interpolation to sound more harmonic, while the inference would be considered more random.
20 samples were generated using the default MuseGAN model. The dimension of the latent space is 128, and is trained using 50000 steps with batch size 64 with an initial learning rate of 0.001. The samples are generated for 4 instruments, of which only the piano will be analysed.

B. Experiment 2: unconditional Transformer and BasicRNN

Both of these models have the option to continue on primer sequences. This option takes a short sequence as input and tries to predict sequences to follow, thus generating a continuation. The two models were used to continue on the 20 samples, described in Subsubsection V-B1. Both were set up to generate 4096 more steps. The temperature of the BasicRNN model was set to 1.0.

1) Primer sequences: From the dataset "Jazz ML ready MIDI" [11] 20 random songs were selected. The 20 samples contained songs such as NY State of Mind by Billy Joel and Celebration by Kool & The Gang. For inputting as primer sequences, all the tracks other than piano were deleted and only the first ten seconds were extracted. These are also the files used for comparison in the results.

C. Experiment 3: influence of temperature on the MusicVAE model

For this experiment, the MusicVAE generated 20 samples per temperature from scratch. The temperature was set to either 0.5, 0.8, 1.0, 1.2 or 1.5. The model used is called the 16 bar hierdec melody model. It is named after the decoders hierarchical architecture. It has a latent space of size 512, built using a bidirectional LSTM encoder of dimension $[2048, 2048]$ and a categorical LSTM encoder. It was trained with a learning rate of 0.001. The decoder consists out of a 4 hierarchically stacked MultiRNN cells which have dimension $[2048, 2048, 2048]$. The temperature of 1.0 will serve as a baseline for this experiment.

VI. Results

The results described below are based on the first samples generated by the models when the experiment was run. None of the samples are curated and the analysis done is based on the 9 features as described in Subsection IV-C. The following tables hold the results of the experiments. The features on the right represent the 9 features proposed by [14]: total pitch count (PC), total note count (NC), pitch class histogram (PCH), pitch class transition matrix (PTM), pitch range (PR), average pitch shift (AvgPS), average inter-onset-interval (IOI), note length histogram (NLH) and note length transition matrix (NLTM).

A. Experiment 1

Sample mode		**Interpolation**	**Inference**
PC	avg	46.4	68.35
	std	33.48	15.59
NC	avg	1538.9	1693.55
	std	540.54	296.18
PCH	avg	0.08333	0.08333
	std	0.08579	0.01981
PTM	avg	43.32	48.11
	std	30.72	22.62
PR	avg	62.3	72.75
	std	23.56	11.21
AvgPS	avg	15.27	13.14
	std	5.10	3.87
IOI	avg	0.05161	0.04617
	std	0.01395	0.009657
NLH	avg	0.08333	0.08333
	std	0	0
NLTM	avg	10.67	11.75
	std	3.753	2.056

TABLE II: Experiment 1: Comparison of MuseGAN's sample modes; inference versus interpolation when generating from scratch, using the 9 key features proposed by Yang et Larch [14]

The results of the measurements of experiment one can be seen in Table II. Fig. 3 and Fig. 4 display the heatmaps of the pitch transitions, for interpolation and inference respectively. All of the notes used in all of the samples, only contained 16^{th} notes. The histograms representing the average pitch counts in the samples are presented in Fig. 5 for interpolation and Fig. 6 for the samples generated with interpolation.

Model / dataset		**Primer**	**Transformer**	**RNN**
PC	avg	19.05	34.2	28.65
	std	7.605	16.32	5.387
NC	avg	93.45	187.95	1555.8
	std	46.63	92.88	1006.5
PCH	avg	0.08333	0.08333	0.08333
	std	0.1005	0.09631	0.06578
PTM	avg	0.05069	0.06805	0.07986
	std	0.1090	0.09677	0.09439
PR	avg	47.85	56.9	31.55
	std	10.07	18.17	4.128
AvgPS	avg	10.65	9.998	3.875
	std	3.447	5.046	1.235
IOI	avg	0.1098	0.1167	0.4931
	std	0.07476	0.0549	0.2646

TABLE III: Experiment 2: Performance of the Transformer model compared to the BasicRNN when continuing primer sequences, using the 9 key features proposed by Yang et Larch [14].

B. Experiment 2

Table III displays the averages of the results of experiment 2. One can see how the average note count for the RNN is many times higher than for the Transformer. While the samples of the Transformer were around 90 seconds long, the files the RNN outputted were around 5 to 9 minutes long. The average pitch count for the Transformer is 34.2, on an average note count of 187.95. It has an average pitch range of 56.9 and its average pitch shift is 9.998. The pitch transition matrices, seen in Fig. 7, Fig. 8 and Fig. 9, are heatmaps representing the pitch transitions of the samples of the primers sequences, the samples generated using the Transformer and the samples generated using the BasicRNN. Figure 10 represents the different note lengths that were used in the samples generated by the Transformer. The note lengths used by the BasicRNN were mostly $8th$ and $16th$ notes.

C. Experiment 3

Temperature		**0.5**	**0.8**	**1**	**1.2**	**1.5**
PC	avg	14.1	13.3	14.5	16.55	15.55
	std	4.998	4.889	4.031	6.028	6.938
NC	avg	76.75	73.25	81.9	74.8	71.65
	std	22.06	23.60	29.39	32.01	24.10
PCH	avg	0.08333	0.08333	0.08333	0.08333	0.08333
	std	0.09882	0.09002	0.081941	0.07774	0.07168
PTM	avg	0.05520	0.05069	0.05972	0.05486	0.05763
	std	0.1278	0.1107	0.1243	0.1142	0.1135
PR	avg	26.45	24.45	22.75	31.2	26.35
	std	13.78	16.61	9.337	17.03	13.45
Avg PS	avg	4.218	3.309	3.9555	4.172	4.317
	std	2.119	1.804	2.4320	1.822	2.315
IOI	avg	0.4620	0.4740	0.4557	0.5070	0.4857
	std	0.1959	0.1251	0.2078	0.2003	0.1350
NLH	avg	0.08333	0.08333	0.08333	0.08333	0.08333
	std	0.06950	0.07208	0.06498	0.06507	0.06915
NLTM	avg	0.52604	0.5017	0.5618	0.5125	0.4906
	std	0.7408	0.7736	0.7997	0.7231	0.7653

TABLE IV: Experiment 3: Comparison of the difference in temperature in the MusicVAE model, using the 9 key features proposed by Yang et Larch [14]

Table IV contains the results obtained from experiment 3. Fig. 12 displays a histogram portraying the average note counts per pitch per temperature. Fig. 13 contains the histograms per temperature for the note lengths. The pitch class transition matrices for temperature 1.0, 0.5 and 1.5 can be found in Fig. 14, Fig. 15 and Fig. 16 respectively.

VII. Discussion

The features that have proven to be the most useful from the 9 features by Yang et Larch [14], are the pitch class transition matrix, the note count and pitch class count histograms and the pitch range. Using the pitch class transition matrix, one can deduct whether samples use the same melodic patterns. The pitch range reveals a lot about 'wide' a sample's melody is. Visualizing how often certain notes and pitches are used, helps a lot for arguing whether samples are alike.

A. Experiment 1

As seen in Table II, the biggest difference between the two models is found in the melodic measurements. The average pitch count for inference is higher than for interpolation, at 68.35 and 46.4 respectively. The standard deviation for the first is way smaller than for the latter, which implies that the difference using interpolation is way bigger in between samples regarding the different pitches used. That inference uses melody more freely, can also be deducted from the pitch range, for which the average is about 10 higher.

When comparing the transition matrix to the average pitch of the inference samples, we notice that the pitch shift is 13.14, which is close to 12, the amount of pitches in an octave. Thus the samples often jump exactly one octave. The fact that the pitch count for the samples generated using inference instead of interpolation, give rise to the suspicion that the grid used when sampling is often drawn on notes such as C♯, D♯ and F♯.

Comparing the heatmap and the pitch count for sampled generated using interpolation, Fig. 3 and Fig. 5, it is surprising how F♯ does not have a clear pattern of being followed by another note, except for itself, although it is the most used note. In contrast, D♯ is often followed by itself and besides that it is one of the preferred successors of both C♯ and B♭.

The note count for samples made with inference, Fig. 6, reveals a preference for natural pitches. Especially E, G and A. All the natural pitches, except for F (F♯ is used instead) are part of the E natural minor scale, which leads one to think that sampling using inference has a preference of playing in E natural.

Rhythmically one would assume that the samples would be more alike. All of the notes used in all of the samples, only contained 16^{th} notes. A tiny difference in the average inter-onset-interval, however, makes it clear that these sample do differ rhythmically.

One can conclude that interpolation using the MuseGAN model generates samples that are melodically less varying, as the three most used notes make up for more than half of the notes played, as seen in Fig. 5. Interpolation has a wider variety of transitions, whereas inference often leads to the one pitch being followed by the same pitch, not regarding

octaves. Rhythmically the sampling modes are much alike, using only 16^{th} notes, but implementing breaks at slightly different times. This raises the suspicion that the sampling mode doesn't change the rhythmical aspects of the generated samples.

B. Experiment 2

Rather non-surprising, it is clear that the Transformer uses quite some more pitches. It is impressive, however, how it uses that much more than the BasicRNN. With an average pitch count of 28.65 on an average note count of 1555.8, it is clear that the BasicRNN uses a lot of repetition. From the pitch transition matrices, Fig. 7, Fig. 8 and Fig. 9 one can spot the similarities between the one of the primer sequences and the one of the BasicRNN. This entails that the BasicRNN mimics a transition more often than the Transformer does. The latter, however, shows the same pattern spotted as in the MuseGAN model, showing a clear preference of following a note with the same note, possibly an octave higher or lower. From Fig. 10, the note length histogram for the samples of the Transformer, one can deduct that it samples use a wide variety of note lengths, having a preference for 16^{th} notes, which is unsurprising given that these notes, apart from being the shortest, usually follow each other up. This is confirmed by inspecting Fig. 11, representing the note length transition matrix of the Transformer. The occurrence of triplets, as well as the dot 8^{th} and dot 16^{th} notes in combination with their regular counterparts, implies that the Transformer is able to sample complex rhythmical structures.
The Transformer clearly generates more interesting samples, but in a comparison of which model continues the primers better, the BasicRNN wins. The samples generated by the Transformer sound good and coherent, while the samples from the RNN are often very repetitive and have a high inter-onset-interval, meaning that there are a lot of silences.

C. Experiment 3

A first thing that comes to the attention when inspecting the results, is that is a hard to find a real connection between the temperature and certain characteristics. Apart from a very similar average note and pitch count, the average inter-onset-intervals are nearly the same. With a pitch range remarkably higher at temperature 1.2, measuring at 31.2, it nears the baseline again at temperature 1.5. Less subtle differences can be found in the pitch class transition matrices, for which the temperature changes the recurrent sweet spot. Temperature 1.0 transitions the most often from C to C, while at temperature 0.5 and 1.5 this changes to G and D respectively. Furthermore, an increase in temperature leads to lower values in the diagonals, implying that an increased temperature highers the chance of a note not being followed by the same note. The note length histogram shows a clear preference for 8^{th} notes when the temperature is 1.0. A change in temperature, most notable at temperature 0.5 and 1.5, leads to more varied rhythmical features, such as in increase in triplets and dot 8^{th} notes. The transition matrix for temperature 1.0 shows a clear preference for following up 8^{th} notes by each other. With a temperature of 0.5, it gives rise to combination of 8^{th} notes paired with both 16^{th} and quarter notes. A temperature of 1.5 leads to samples with a higher rate of 16^{th} notes. From the transition matrix one can clearly notice an increase in the dot 8^{th} notes followed by 16^{th}, a more complex rhythmical structure.
The temperature used when sampling from MusicVAE, has a subtle difference on the output. The higher the temperature, the higher the complexity of the rhythmical features. Furthermore, it yields a higher chance of notes not being followed by each other, thus acquiring a higher variety in the pitch intervals.

VIII. Limitations

A. Training of the models

Training of these complex models is both computationally heavy and resource intensive, therefore it was not possible to train models on the same datasets during the scope of this project. Many attempts were made to train the different models, namely multiple GAN models and Transformers, but without any good results. On small datasets, it was achieved to train the MusicVAE and RNN models, but they were no fit comparison to each other, since one is mainly used to generate from scratch and the other for continuation on primer sequences.
This limitation led to a change in the scope of the project. Originally, the goal was to make an objective comparison of the different models when used to generate from scratch. The inability to train the models on the same dataset, would lead to an unfair comparison.

B. Ambiguous information on the training data

For all models described in this work, no or ambiguous information on the training data was provided. In order to have an objective measurement for our evaluation method used, generated samples should be compared to the dataset they were trained on.

C. Evaluation method limited to one instrument

Unfortunately, the evaluation method implemented only works with one instrument: piano. The functions could be rewritten to work for multiple elements, but the scope of this project, of which the main focus is piano music, did not include further adaptation of the evaluation method.

IX. Conclusion

In this work an analysis of different music generating models and their setups has been made. The limitations mentioned in Section VIII, lead to results that are not measurable when comparing different models to each other.
Samples generated by the MuseGAN are rhythmically equal to each other, regardless whether these samples were generated using interpolation or inference. Inference yields samples with more melodic variety, although the same notes are more often played after one another. The samples have a significantly

higher concentration of natural tones and often follows the E natural scale. Interpolation generates three certain notes, C♯, D♯and F♯, a lot more often than all the others. Of these notes, the last one is often used for transitioning to other pitches, while the first two are, especially D♯, are usually played back to back.

When continuing on primer sequences, the Music Transformer generates samples that are more complex and interesting compared to the BasicRNN. The BasicRNN implements a lot of breaks, but its output has more similarities to the primer sequence than the Transformer's samples.

Samples generated from scratch using the MusicVAE, are influenced by the temperature with which these were generated. The higher the temperature, the less the samples contain pitches transitioning into itself. An increase in temperature also gives rise to more complex rhythmical features, such as the use of triplets and the combination of using dotted notes together with their regular counterparts.

Of the 9 features, the most expressive in terms of comparison were the pitch class transition matrix, the note count and pitch class count histograms and the pitch range.

To form an objective comparison of the models presented in this work, future work consists out of training the models on the same dataset and using the measures of evaluation to calculate a scalar, measuring the inter-set and intra-set distances between samples generated and the dataset on which the models would be trained. Furthermore, elements such as chord progression could be implemented such that one could generate tracks, or accompaniment, based on a preexisting chord progressions used in projects. This would allow these techniques to be used a tools for musicians.

BIBLIOGRAPHY

[1] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein gan, 2017.

[2] S. R. Bowman, L. Vilnis, O. Vinyals, A. M. Dai, R. Józefowicz, and S. Bengio. Generating sentences from a continuous space. *CoRR*, abs/1511.06349, 2015. URL http://arxiv.org/abs/1511.06349.

[3] J.-P. Briot, G. Hadjeres, and F.-D. Pachet. Deep learning techniques for music generation – a survey, 2019.

[4] H.-W. Dong, W.-Y. Hsiao, L.-C. Yang, and Y.-H. Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[5] I. J. Goodfellow. NIPS 2016 tutorial: Generative adversarial networks. *CoRR*, abs/1701.00160, 2017. URL http://arxiv.org/abs/1701.00160.

[6] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, A. Dai, M. Hoffman, M. Dinculescu, and D. Eck. Music transformer: Generating music with long-term structure. 2019. URL https://arxiv.org/abs/1809.04281.

[7] A. D. Nunzio. Illiac suite, 2011. URL https://www.musicainformatica.org/topics/illiac-suite.php.

[8] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, Feb 2020. ISSN 1433-3058. doi: 10.1007/s00521-018-3758-9. URL https://doi.org/10.1007/s00521-018-3758-9.

[9] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit. A decomposable attention model for natural language inference. *CoRR*, abs/1606.01933, 2016. URL http://arxiv.org/abs/1606.01933.

[10] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. A hierarchical latent vector model for learning long-term structure in music, 2019.

[11] Sai. Jazz ml ready midi, 2018.

[12] P. Shaw, J. Uszkoreit, and A. Vaswani. Self-attention with relative position representations. *CoRR*, abs/1803.02155, 2018. URL http://arxiv.org/abs/1803.02155.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

[14] L.-C. Yang and A. Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 32(9):4773–4784, May 2020. ISSN 1433-3058. doi: 10.1007/s00521-018-3849-7. URL https://doi.org/10.1007/s00521-018-3849-7.

[15] L.-C. Yang, S.-Y. Chou, and Y.-H. Yang. Midinet: A convolutional generative adversarial network for symbolic-domain music generation, 2017.

X. APPENDIX

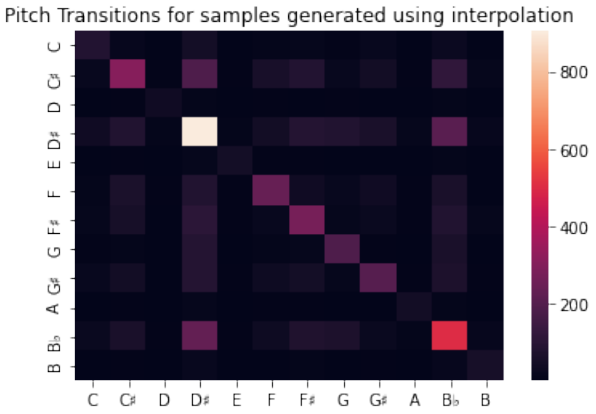

Fig. 3: Heatmap of the pitch transitions from samples generated using MuseGAN's interpolation

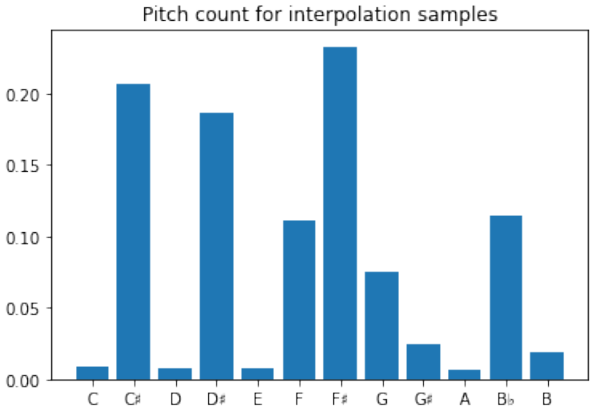

Fig. 5: Histogram of the pitch occurences in MuseGAN's interpolation samples

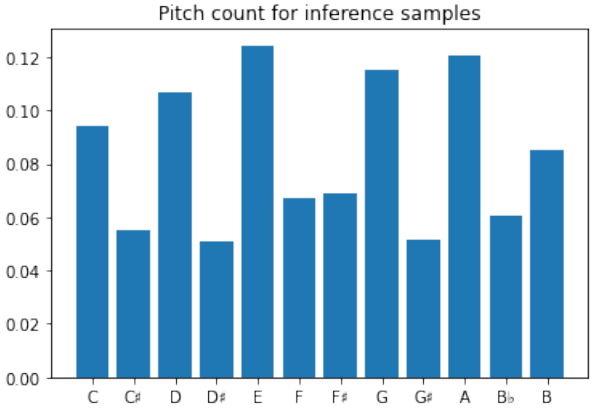

Fig. 6: Histogram of the pitch occurences in MuseGAN's inference samples

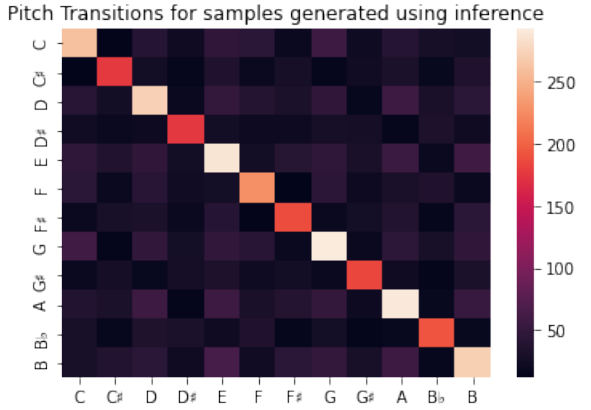

Fig. 4: Heatmap of the pitch transitions from samples generated using MuseGAN's inference

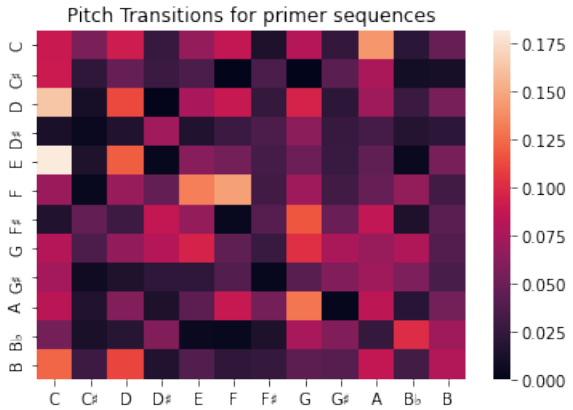

Fig. 7: Pitch Transition matrix of the primer sequences

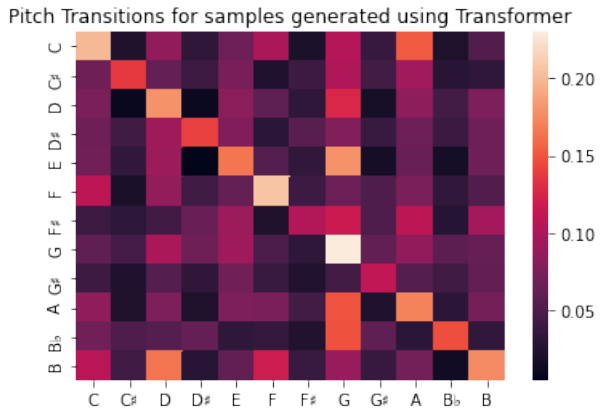

Fig. 8: Pitch Transition matrix of the samples generated with the Transformer model

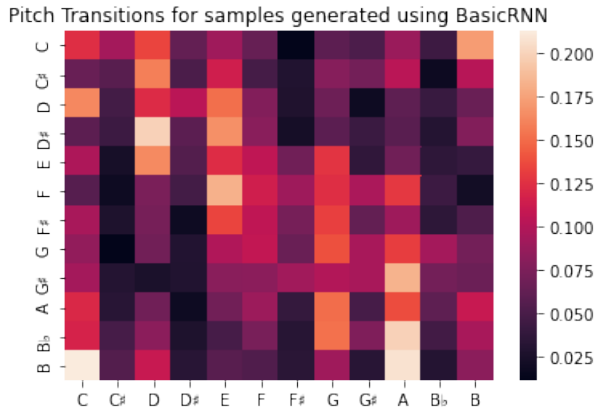

Fig. 9: Pitch Transition matrix of the samples generated by the BasicRNN

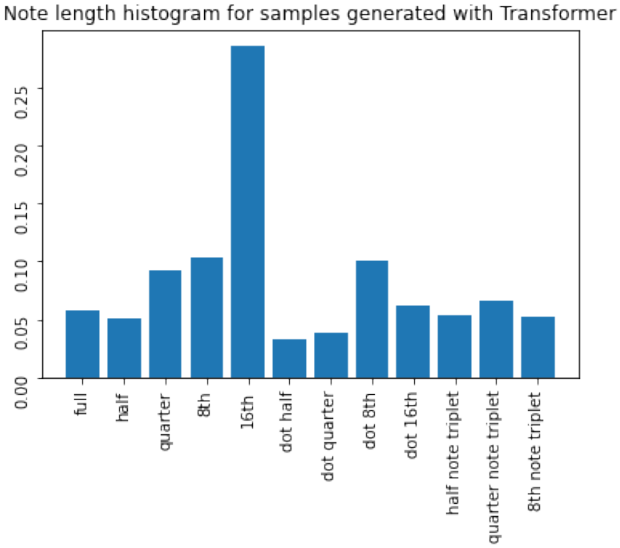

Fig. 10: Note Length histogram of the samples generated with the Transformer model

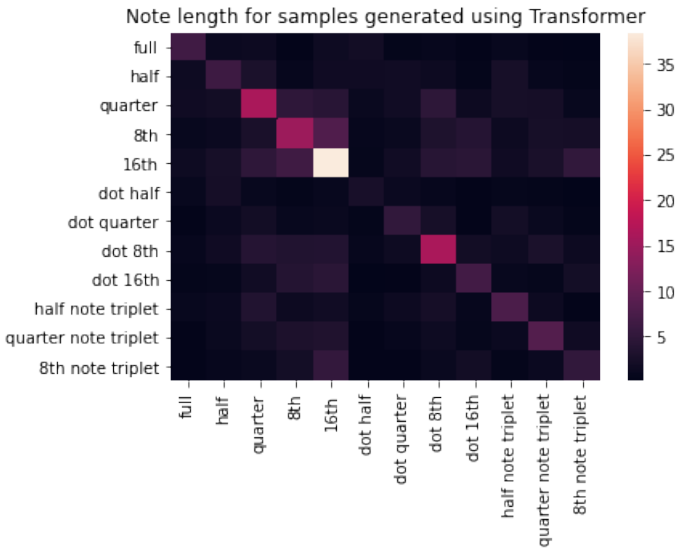

Fig. 11: Note Length Transition matrix of the samples generated with the Transformer model

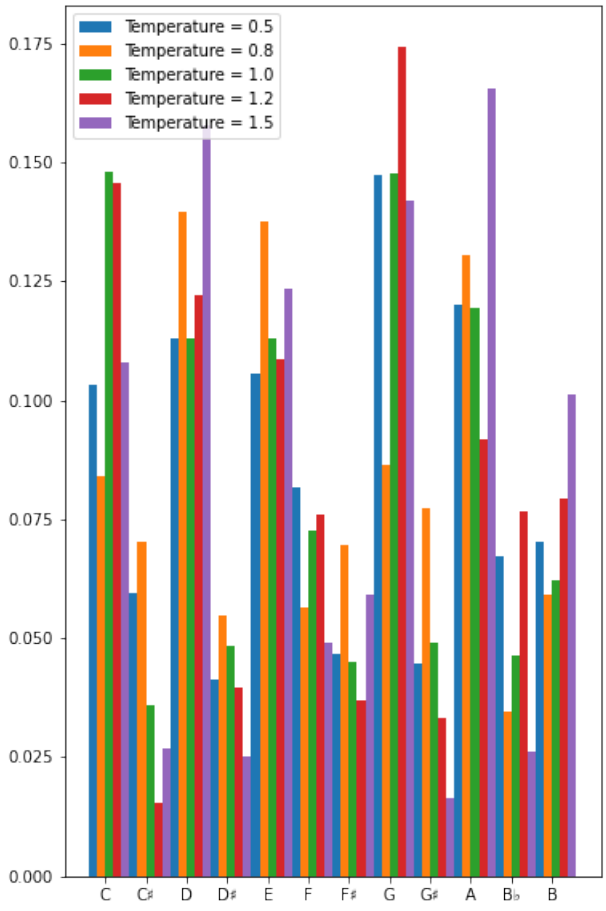

Fig. 12: Histogram containing the average counts per pitch per temperature for experiment 3

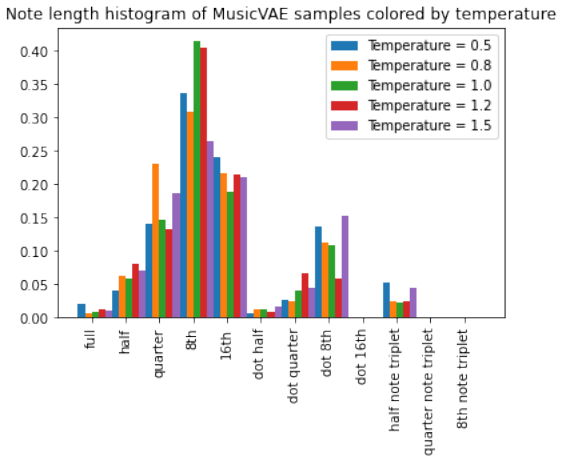

Fig. 13: Histogram containing the average counts per note length per temperature for experiment 3

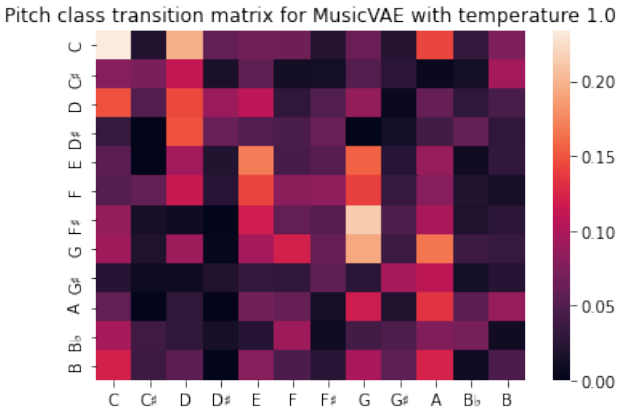

Fig. 14: Pitch class transition matrix for samples from the MusicVAE model with temperature 1.0

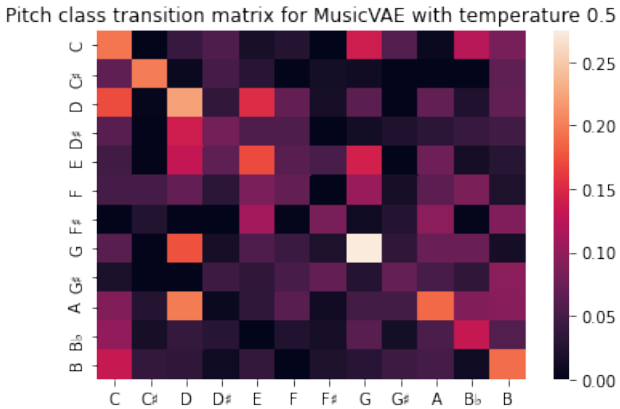

Fig. 15: Pitch class transition matrix for samples from the MusicVAE model with temperature 0.5

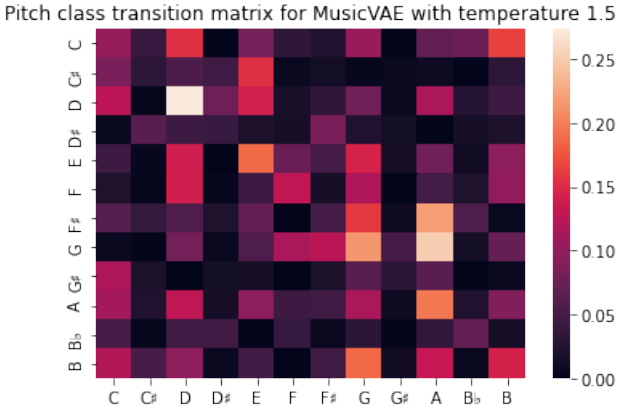

Fig. 16: Pitch class transition matrix for samples from the MusicVAE model with temperature 1.5

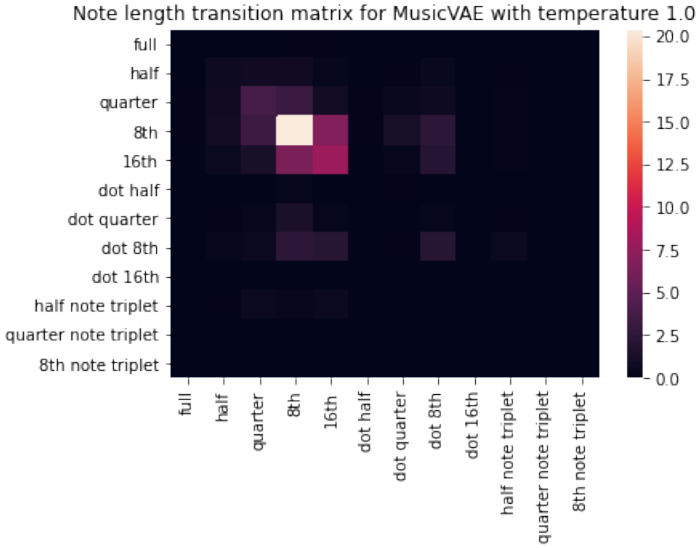

Fig. 17: Note length transition matrix for samples from the MusicVAE model with temperature 1.0

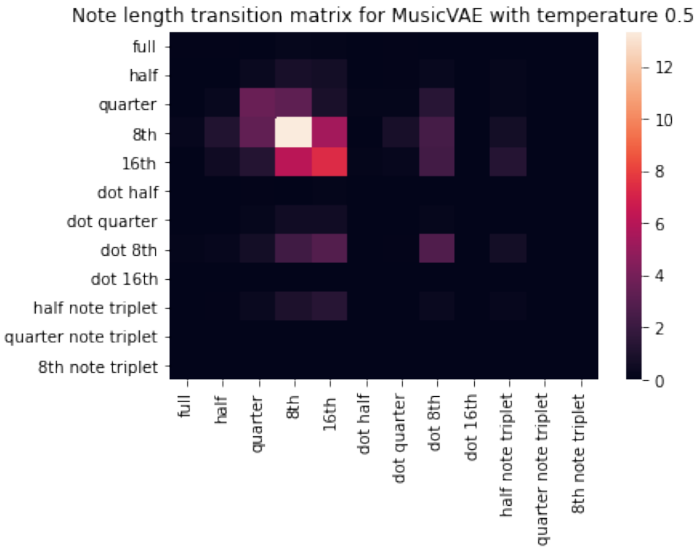

Fig. 18: Note length transition matrix for samples from the MusicVAE model with temperature 0.5

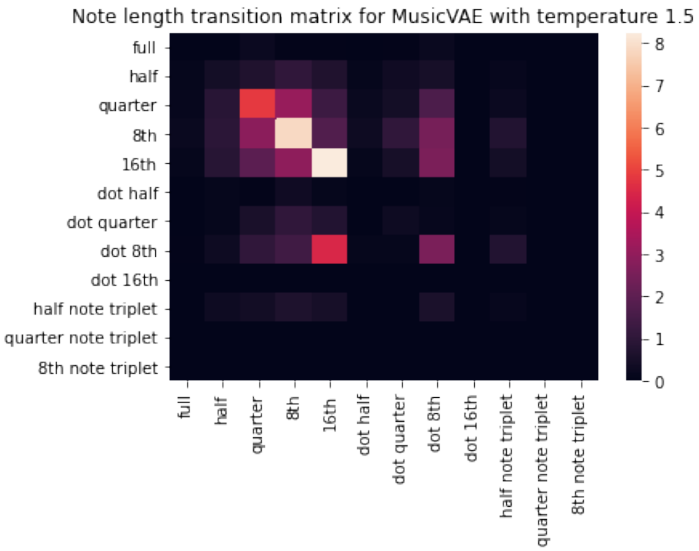

Fig. 19: Note length transition matrix for samples from the MusicVAE model with temperature 1.5

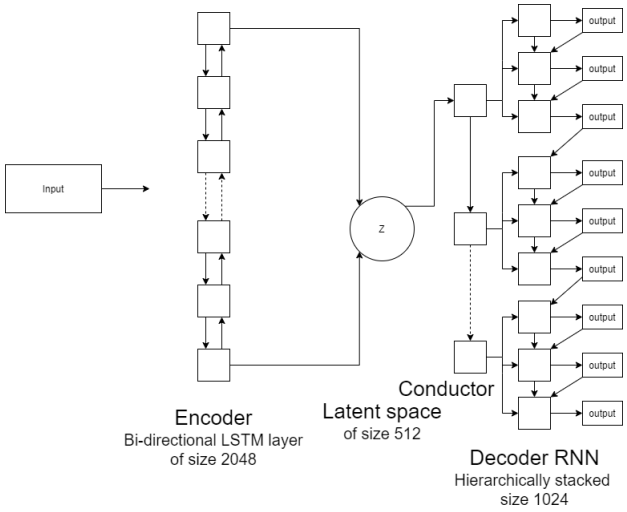

Fig. 20: A visualisation of the MusicVAE model

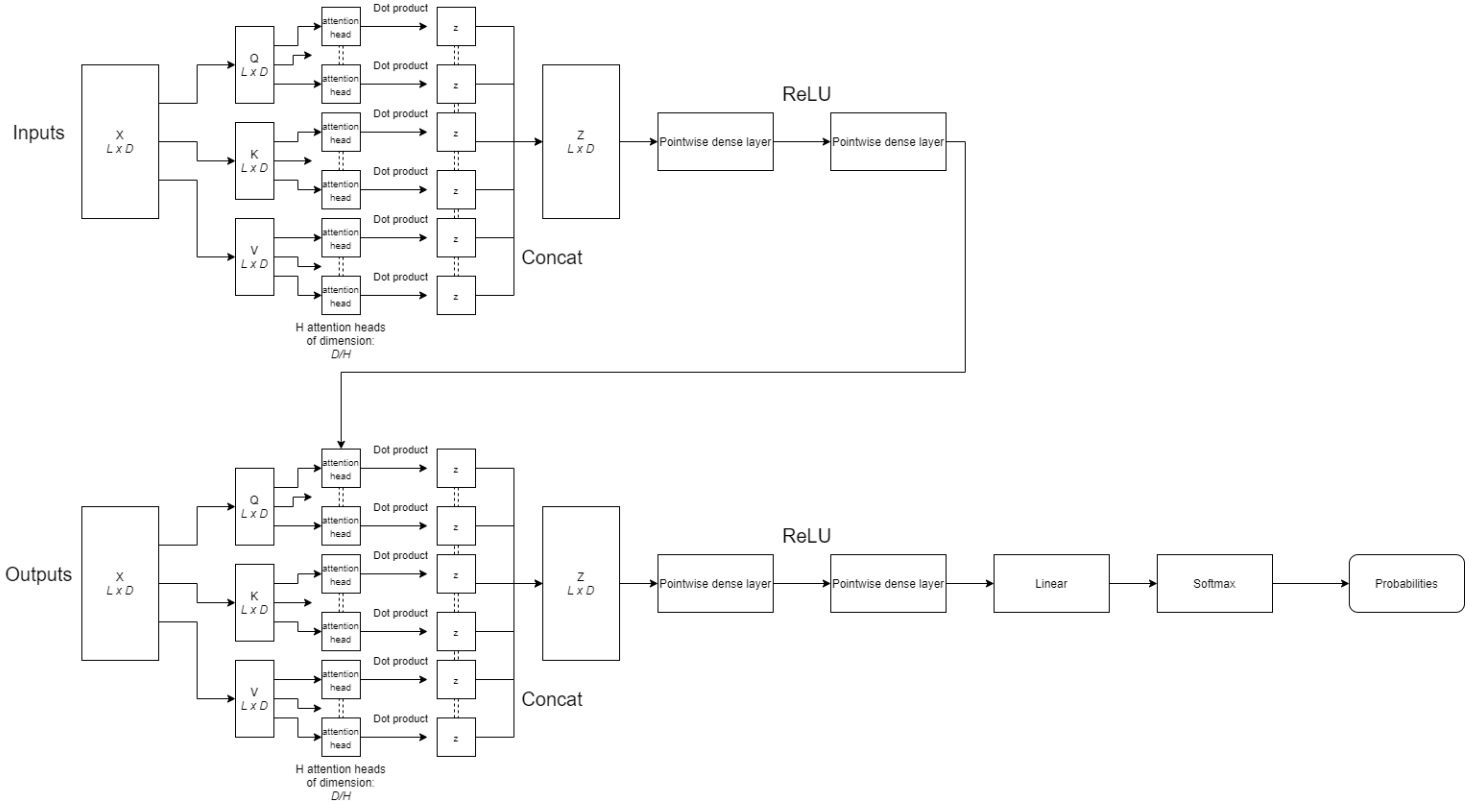

Fig. 21: A visualisation of the Transformer model

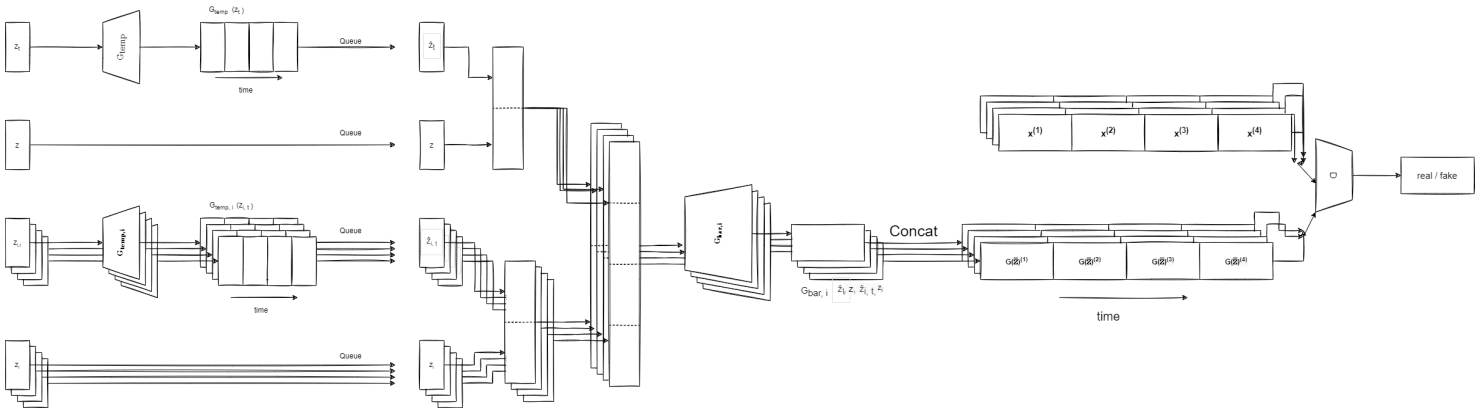

Fig. 22: A visualisation of the MuseGAN model