Transformer Neural Networks for Automated Rhythm Generation

Thomas Nuttall

Supervisor: Sergi Jordà
Co-Supervisor: Behzad Haki

September 2020
Transformer Neural Networks for Automated Rhythm Generation

Thomas Nuttall

Supervisor: Sergi Jordà
Co-Supervisor: Behzad Haki

September 2020
Contents

1 Modelling and Rhythm Generation ................................................................. 1
  1.1 Related Work ......................................................................................... 2
    1.1.1 Early Computational Methods ......................................................... 2
    1.1.2 Neural Networks for Automated Composition ................................. 3
  1.2 Objectives and Expectations ................................................................. 9

2 Architecture ................................................................................................. 10
  2.1 Groove MIDI Dataset ............................................................................ 10
  2.2 Sequence Tokenisation .......................................................................... 11
  2.3 Modelling ............................................................................................. 17
    2.3.1 Transformer-XL Model ................................................................... 18
  2.4 Sampling and Generation ....................................................................... 19
    2.4.1 Task 1: Generation .......................................................................... 19
    2.4.2 Task 2: Continuation ...................................................................... 20
  2.5 Evaluation in Development ................................................................... 20
    2.5.1 Perplexity ....................................................................................... 21

3 Evaluation, Experimentation and Analysis .................................................... 22
  3.1 Offline Evaluation - Model Perplexity .................................................... 22
  3.2 Online Evaluation - Listening Experiments .......................................... 23
    3.2.1 Test Samples .................................................................................. 23
    3.2.2 Experiment Setup .......................................................................... 24
    3.2.3 Results .......................................................................................... 24
Dedication

To the Mighty Quinn, may this serve as a baseline for finding your own groove in life.
Acknowledgement

Naturally I owe a big bag of thanks (one each) to Sergi and Behzad for the ongoing supervision, advice and good-nature offered to me over the course of this project. It is a regret that we didn’t get to meet in person as much as I would have liked.

More broadly - what with this work marking the end of a unique and fascinating journey through sound and space - I would like to thank Miguel for being a valuable friend and collaborator during these two years, on the pitch and off.

And finally I express gratitude to the authors of the LakhNES paper, whos ISMIR presentation inspired me to learn, investigate, enjoy and to some extent even succeed in the application of Transformers in the music domain... particularly to Chris Donahue who was kind enough to answer some of my ad-hoc questions and provided some key bits of code for the project.

Also big respect to my Mum even though she doesn’t really have any idea what I am doing.
Abstract

We propose a novel approach to automated rhythm generation in which a Transformer-XL model is employed to model and generate rhythm from the Magenta Groove MIDI Dataset. Recent applications of this high-dimensional language framework in the field of music have demonstrated it’s ability to effectively capture and emulate long-term dependency in musical sequences - dependencies characteristic of human notions of musicality and creative merit - making it an ideal candidate to experiment with for the task of rhythm-specific generation. We evaluate hundreds of generations from our optimum model using a variety of methods; probabilistic, musicological and in blind listening tests to determine the extent to which our framework has learnt and reproduced the aspects of rhythm we understand to be valuable. Our model is able to achieve a standard of rhythmic production comparable to human playing across arbitrarily long time periods and multiple playing styles.

Keywords: Transformer-XL, Automated Music Generation, Rhythm Generation
Chapter 1

Modelling and Rhythm Generation

In this paper we are concerned with musical rhythm, definitions for which are numerous and varied but all point towards a core concept of repetition, particularly of patterns in sound.

It should be obvious that where patterns exist, we can to some extent capture numerically - in fitting some model to example data - intrinsic properties of those patterns, and hence rhythm. What might not be so obvious is why it is important to, or what can be achieved when we do so.

Translating rhythm from performance to numbers in a statistical model can be thought of as a process of dimensionality reduction, we capture as much rhythmic essence as possible and project it into a lower dimensional, machine friendly space. From a musicological perspective, this common abstract projection provides a shared language to describe, document, teach and transmit rhythmic structure. With the reduction in complexity providing an easier method to store, compare and explore it analytically - analysis’ such as performance evaluation or quantification of creativity [1].

Though perhaps most interesting of all in the abstraction of rhythmic input into this domain is the learning of a foundation from which we are able to create, generate or even compose new sounds whilst maintaining the spirit of the original rhythm. Not
only is this exciting from an artistic viewpoint, it too provides us with a method by which we are able to evaluate our rhythmic representation - if we can use our learning to produce new sounds and new rhythms, we can hold our model to account just as we can any performance artist.

This process of abstraction, optimisation, reproduction and evaluation builds on a long history of increasingly more sophisticated techniques in algorithmic musical composition.

1.1 Related Work

The task of algorithmic composition in music is not new. In fact musicians have been attempting to create non-deterministic rule-based compositions for centuries. From Mozart’s utilization of dice to randomly dictate the order in which his musical segments were stitched together in his Musikalisches Würfelspiel to John Cage’s Reunion, composed whilst performed with sounds triggered by ongoing games of chess [2, 3]. Unfortunately the vast majority of works in the field are concerned with pitch (rather than strictly rhythm), as such it would be remiss not to leave these out of the discussion.

1.1.1 Early Computational Methods

In more recent times, computers have been the tool of choice for automated music generation, bringing with them huge (and ever developing) advances in complexity and capability. Stochastic systems such as that outlined by Iannis Xenakis’ in Formalised Music [4] relied on digital random-number generators to achieve what Cage and Mozart did before him whilst Anderson et al. rely on Markov Chains to realize a more advanced level of stochasticity [5]. Conversely, projects like MUSICOMP [6] or William Shottstaedt’s Automatic Counterpoint [7] relied on rule-based approaches in which, once initialised and set in motion, a piece behaves according to a set of rules prescribed by the designer [3]. An extension (in principle) to rule-based compositional techniques is the current state-of-the-art in automated music generation - artificial intelligence, or more specifically (at the time of writing), neural networks.
The distinction in these modern systems is that they are able to learn their own rules to a level almost inconceivable to humans and hence have the ability to generate sound much more diverse and complex than any of their predecessors.

1.1.2 Neural Networks for Automated Composition

Neural networks (NN’s) used for music generation in recent years come in a variety of flavours with difficulty in evaluating them objectively and a wide range of use cases making it hard to single out a single one as best.

Recurrent Neural Networks

One early variety of NN able to model time-series input are recurrent neural networks (RNN’s) - feed-forward neural networks extended to ingest sequential data by including recurrent connections. At each time step, the input of the RNN is an element of the sequence and that of the one before it - that is to say that each element is recursively passed back to coincide with the one succeeding it. The expected output is the next element in the sequence. Hence we have a network that learns to predict the next step in a sequence using its current state, the one before it, and, therefore, in theory, all preceding states [2, 8].

Recurrent layers can be conceptualised as occurring multiple times, the step of feeding the sequence back into itself can be pictured as feeding the sequence into another identical network with same weights who’s learning is linked to previous the previous one. Figure 1 illustrates this, on the left is the recurrent layer, on the right is the layer unrolled. Each module in a recurrent neural network makes some
transformation to its input data, in the case of RNNs, this transformation is quite simple, like for example a tanh as shown in Figure 2.

![Figure 2: Repeating module of standard RNN](image)

One can see how this architecture is a natural choice for music where the sequential nature of the data is explicitly learned by the model. Efforts in algorithmic compositions using RNN’s were made as early as the 80s [10], though a more recent notable attempt can be found in [11] where a network of restricted boltzmann machines is used to learn harmonic and rhythmic probabilistic rules from polyphonic music scores. Both examples note difficulty in effectively learning/generating convincing long-term structure, with generated musical lines being impressive, but short. This inability of RNN’s to learn effectively in the long-term is due to how the loss function is defined - since the gradient decays exponentially in time, long-term dependencies are hidden [12]. This should not be hard to appreciate by looking at Figures 1 & 2 - there is no explicit mechanism to identify where or how far back in the sequence relationships exist, the setup is such that the further back in time we go, the less relevant the relationships are assumed to be. This obviously poses a problem for modelling music where relationships exist periodically over the entirety of the data and is addressed by long-term short-memory networks (LSTMs).

**Long-Term Short-Memory Networks LSTM**

LSTMs are a variation on the traditional RNN with additional special units introduced to maintain information in memory for longer time periods, with gates
1.1. Related Work

Figure 3: Repeating module of standard LSTM [9]

determining which information is stored and for how long (in contrast to vanilla RNN’s which replace the activation at each time step). Figure 3 shows how these gates are configured in the repeating module of an LSTM. The path across the top of the module is the cell state, this is largely unaffected by transformations on its own path but can have information added or removed by the gates feeding into it (the sigmoid neural network layer components). Gates output a number between 0 and 1 representing the extent of information let through and in doing so inform the cell state on what is important to remember at each time step in relation to other steps in the cell state (how many steps are considered in one hidden state is a parameter of the model and directly dictates the LSTMs ability to learn in the long-term). This gate improvement not only enhances the network’s ability to preserve long-term temporal information, but also proves to be more computationally efficient [13].

One of the first notable applications of LSTM networks to the task of automated composition was in [14] where Eck et al produce a remarkable polyphonic blues improvisation using a LSTM in which chord and melody connections are decoupled, with chords influencing melody but not vice-versa. A novel representation of musical
sequence is used in [15] where chords are represented as strings of text in the input sequence rather than musical entities (such as [C, :, m, a, j, G, :, m, a, j] rather than [C:maj, Gmaj]), observing that fewer existent characters in the input space reduced complexity and increased runtime in train - they did however note that this method failed when applied to percussion. Transposition invariance is achieved in [16] by tying together multiple parallel LSTM’s (by which they mean coupling the weights of each network), one for each note in the input sequence and also in [17] by representing input data as a two-dimensional matrix of playable notes and time with a supplementary matrix indicating whether a note is repeated, held or neither. The Magenta project boasts expressive timing and dynamics with LSTM’s in [18] by fixing the time “step” to 10ms and allowing the model to skip forward in time if necessary (rather than tying the time step to the meter). There is limited contribution to the strictly rhythmic domain with LSTM’s, though [19, 20] both achieve low-dimensional generation (3 distinct sounds) on limited data. Two operational consequences of the increased complexity associated with LSTM’s gated memory functionality are that they exhibit longer, more memory intensive training times and that they are more prone to overfitting (as a consequence of increase in parameters). From a model performance perspective, they are still not considered perfect in how they model temporal dependency in that there is still an emphasis on proximity in the input sequence, one imagines that in many cases this is important musically but this does come as a trade off with emphasis on distant relationships.

**Attention-based Mechanisms**

These drawbacks of LSTM networks are problems addressed by a more recent architecture that too is concerned with long-term memory, that is, self-attention, first proposed in [21]. At a high-level, the self attention - or more commonly, attention - mechanism achieves what gated memory does in LSTM’s by learning a self-similarity matrix between the input sequence and itself. This matrix is easily conceptualised by thinking of a square similarity matrix where each element of the sequence is a row or column on the grid, the diagonal taking the maximum value - most similar/relevant. An example with a text sequence can be found in Figure 4 which displays the
1.1. Related Work

Figure 4: Attention Matrix for Translation Task [21]

relationship between text tokens in a translation task that have been learnt via an attention mechanism.

Learning the temporal dependencies in input sequence using this self-similarity approach removes the relationship between importance and proximity - any element in the sequence can be identified as relevant to any other within the constraints set by the dimensions of the grid, this is a hyperparameter of models that use these mechanisms and can be thought of as analogous to the size of the hidden cell state in LSTMs (we will talk more about how this constraint is removed by our model of choice later on). Early descriptions of the mechanism can be found in [21, 22, 23].

Attention is at the heart of the transformer neural network, introduced in 2017 [24]. Transformers typically consist of an encoding module of feed-forward neural networks (FNN’s) which take as input sequential data and produces some embedding for each element. Self-attention is applied, aggregating information from all other elements, generating a new per-element representation learned from the entire context - this is repeated to create successive generations of more nuanced embeddings. A decoder then generates an output sequence element by element while consulting the representation generated by the encoder [24, 25]. Figure 5 illustrates the basic units of a transformer (of which there are many stacked). The attention layer here
replace the recursive mechanism present in the previous two networks.

Due to the use of FNN’s and the lack of recursion (as in RNN’s and LSTM’s), computation can be done in parallel, making transformers much faster to train than their predecessors. In machine translation and NLP tasks, transformers have achieved state-of-the-art in recent years (for example in [26]) though even more recently have been applied to the task of music generation. Google was first with their Music Transformer in early 2019, showcasing state-of-the-art perplexity on the scale of minutes (as opposed to seconds as before) and demonstrate the ability to expand on a musical prime as input, noting that they foresee the how this might be useful as a creative tool [27]. The MuseNet project from OpenAI have also succeeded in generating music at the minute-scale using transformers and a novel input representation in which each token is encoded to combine pitch, volume and instrument [28]. Transformers have also been shown to do well in multi-instrumental applications for example in [28]. Also notable in [28] is the use of the same pre-training method introduced in [26] where learning is performed on a large dataset to understand general musical structure and then fine-tuned on a smaller dataset to capture more style specific relationships. They also utilise the transformer XL implementation [29] which learning of the input sequence to take place across arbitrary length.
1.2 Objectives and Expectations

Although still a hotly developing field, there exists very little work dedicated solely to the modelling and generation of rhythm or percussion, the recently published Groove Midi dataset \[30\] provides a decent source of symbolic rhythmic data to work with, as is done in the accompanying paper \[31\] where LSTM’s are used to infill a low-dimensional input sequence with more complex rhythms. Although impressive, one of the drawbacks stated in the paper is the inability and computational expensiveness of learning in the long-term. We believe that a Transformer architecture (particularly Transformer-XL) can offer a solution to this, producing similarly interesting results with a better understanding of long-term dependency. For this we investigate the Groove MIDI dataset, train an optimum Transformer model and assess it’s ability to learn and emulate the musicality of human rhythm across a variety of styles.

Our output will be a trained machine learning model with an interface to (1) generate new rhythms at user specified tempo and (2) continue a user defined input rhythm (both seen and unseen at training time). We will subject our model to a series of tests and analysis’ to determine the extent to which it has effectively modelled consonant, interesting and musically valuable rhythm as we understand it. These tests include (1) a probabilistic evaluation of the models ability to predict rhythm in a sequence (2) Turing Listening Tests to determine the models ability to imitate human drumming and (3) a musicological analysis on a sample of outputs to place the models contributions amongst those that exist from non-machine methods.

The task we present here has not been achieved to date and we hope that our efforts contribute to the ongoing body of work committed to the development of computational creativity and the statistical modelling of music and sound, grateful so we are for the countless contributions on which this work is built.
Chapter 2

Architecture

The project pipeline is constructed using the code found in the Transformer-XL repository as a template [29]. All configuration and model hyperparameters were abstracted to configuration files and experiments in data representation/model training were carried out on two NVIDIA K80 GPUs to iterate towards our best model (achieved after approximately 4 hours of training). The code and instructions on how to use it can be found at:

https://github.com/slimranking/bumblebeat/tree/master/bumblebeat

2.1 Groove MIDI Dataset

We rely on the Magenta Groove MIDI Dataset (GMD) for training - 13.6 hours (22,000 measures) of MIDI of human-performed, tempo-aligned expressive drumming played mostly by professional drummers. The data is already split into train, test and validation sets which we will use here so as to make comparison with other works more tractable (see Table 1). [30]

All samples are matched with associated metadata including anonymized drummer identifiers, musical style annotations, and tempo. Almost all samples are played in

https://github.com/kimiyoung/transformer-xl/tree/master
2.2. Sequence Tokenisation

| Split   | Beats | Fills | Measures (approx.) | Hits   | Duration (minutes) |
|---------|-------|-------|--------------------|--------|--------------------|
| Train   | 378   | 519   | 17752              | 357618 | 648.5              |
| Validation | 48   | 76    | 2269               | 44044  | 82.2               |
| Test    | 77    | 52    | 2193               | 43832  | 84.3               |
| **Total** | **503** | **647** | **22214**         | **445494** | **815.0**         |

Table 1: Train, Test and Validation Splits of GMD

| Genre     | Count | Proportion |
|-----------|-------|------------|
| rock      | 341   | 0.297      |
| funk      | 160   | 0.139      |
| jazz      | 101   | 0.088      |
| latin     | 97    | 0.084      |
| hip hop   | 95    | 0.083      |
| soul      | 63    | 0.055      |
| afrocuban | 60    | 0.052      |
| punk      | 58    | 0.050      |
| new orleans | 53 | 0.046      |
| country   | 29    | 0.025      |
| pop       | 27    | 0.023      |
| reggae    | 20    | 0.017      |
| gospel    | 19    | 0.017      |
| afrobeat  | 13    | 0.011      |
| dance     | 7     | 0.006      |
| blues     | 4     | 0.003      |
| highlife  | 2     | 0.002      |
| middle eastern | 1 | 0.001      |
| **Total** | **1150** | **1.00**   |

Table 2: Genre Distribution of GMD

4/4 timing though there are some exceptions. Table 2 presents the distribution of playing style, or genre, across the dataset.

2.2 Sequence Tokenisation

The original MIDI representations can be thought of as a sequence of triples, each element provides a value for pitch, velocity and start time, see equation 2.1. Since we are dealing with rhythm only, duration is irrelevant and end time is discarded.
Important to note that all sequences were quantized to 1/16\(^{th}\)s before training.

\[
MIDI = \left[ (p_1, v_1, t_1), \ (p_2, v_2, t_2), \ ..., \ (p_N, v_N, t_N) \right] \text{ for } n \in [1..N] \quad (2.1)
\]

\(N\) = Number of notes in sequence
\(p_n\) = Pitch of \(n^{th}\) note
\(v_n\) = Velocity \(n^{th}\) note
\(t_n\) = Start time of \(n^{th}\) note

Our desired format is a stream of tokens i.e. a one dimensional sequence representing all samples, their velocities, pitches and temporality. To achieve this we apply three transformations to the MIDI representation above.

1. Pitch Mapping

The Roland TD-11 drumkit that the dataset was collected on records 22 distinct pitches. Many of these pitches are very sparse in the dataset and can be naturally grouped, lowering the dimensionality of the input data and reducing the complexity for the model.

The grouping of pitches we adopt is almost identical to that used in Gillick et al. in [31] (see Table 3). Applying this to the entire dataset reduces it to consist of 9 unique pitches in total; kick drum, snare drum, closed hi-hat, open hi-hat, low tom, mid tom, high tom, crash cymbal, ride cymbal. After applying the mapping, our sequence can be described by equation 2.2

\[
seq = \left[ (m_1, v_1, t_1), \ (m_2, v_2, t_2), \ ..., \ (m_N, v_N, t_N) \right] \text{ for } n \in [1..N] \quad (2.2)
\]

\(m_n\) = Mapped pitch of \(n^{th}\) note
### 2.2. Sequence Tokenisation

#### Table 3: Pitch Mappings of Our Dataset

| Pitch | Roland Mapping | General MIDI Mapping | Our Mapping |
|-------|----------------|----------------------|-------------|
| 36    | Kick           | Bass Drum 1          | Bass (35)   |
| 38    | Snare (Head)   | Acoustic Snare       | Snare (38)  |
| 40    | Snare (Rim)    | Electric Snare       | Snare (38)  |
| 37    | Snare X-Stick  | Side Stick           | Snare (38)  |
| 48    | Tom 1          | Hi-Mid Tom           | High Tom (50)|
| 50    | Tom 1 (Rim)    | High Tom             | High Tom (50)|
| 45    | Tom 2          | Low Tom              | Low-Mid Tom (48)|
| 47    | Tom 2 (Rim)    | Low-Mid Tom          | Low-Mid Tom (48)|
| 43    | Tom 3 (Head)   | High Floor Tom       | High Floor Tom (45)|
| 58    | Tom 3 (Rim)    | VibraSlap            | High Floor Tom (45)|
| 46    | HH Open (Bow)  | Open Hi-Hat          | Open Hi-Hat (46)|
| 26    | HH Open (Edge) | N/A                  | Open Hi-Hat (46)|
| 42    | HH Closed (Bow)| Closed Hi-Hat        | Closed Hi-Hat (42)|
| 22    | HH Closed (Edge)| N/A                | Closed Hi-Hat (42)|
| 44    | HH Pedal       | Pedal Hi-Hat         | Closed Hi-Hat (42)|
| 49    | Crash 1 (Bow)  | Crash Cymbal 1       | Crash Cymbal (49)|
| 55    | Crash 1 (Edge) | Splash Cymbal        | Crash Cymbal (49)|
| 57    | Crash 2 (Bow)  | Crash Cymbal 2       | Crash Cymbal (49)|
| 52    | Crash 2 (Edge) | Chinese Cymbal       | Crash Cymbal (49)|
| 51    | Ride (Bow)     | Ride Cymbal 1        | Ride Cymbal (51)|
| 59    | Ride (Edge)    | Ride Cymbal 2        | Ride Cymbal (51)|
| 53    | Ride (Bell)    | Ride Bell            | Ride Cymbal (51)|
2. Velocity Representation

Our velocity values, \( v_n \), lie in the range \([0, 127]\), these are bucketed to fall within \( B \) equally spaced bins.

\[
\text{seq} = [(m_1, b_1, t_1), (m_2, b_2, t_2), ..., (m_N, b_N, t_N)] \text{ for } n \in [1..N] \tag{2.3}
\]

\( b_n \) = Bucketed velocity of \( n^{th} \) note in \([1..B]\)

for \( B = 2 \):

\[
b_n = \begin{cases} 
1, & \text{if } v_n \in [0, 64] \\
2, & \text{if } v_n \in (64, 127]
\end{cases} \tag{2.4}
\]

for \( B = 3 \):

\[
b_n = \begin{cases} 
1, & \text{if } v_n \in [0, 42.33] \\
2, & \text{if } v_n \in (42.33, 84.67] \\
3, & \text{if } v_n \in (84.67, 127]
\end{cases} \tag{2.5}
\]

\( B \) was chosen by subjective evaluation of the model output at various values. Our approach was to reduce \( B \) from \( B = 10 \) until we found a bucketing with which most buckets were occupied/being generated into a large proportion of the time. This way we avoid needless added complexity for the model to learn or take into account yet maintain sufficient variation in intensity to be interesting. We found 4 a nice balance, this is also in line with the number of choices one might be provided on a more basic drum instrument/software (silence, low, medium, high).

Finally, every \((\text{pitch } m_n, \text{velocity bucket } b_n)\) combination is assigned a unique token corresponding to that pair. With \( B = 4 \) and 9 pitch classes, we have 32 \((9 \times 4)\) unique tokens corresponding to every possible combination of \((m_n, b_n)\). This is
2.2. Sequence Tokenisation

quite an important decision in the data representation, and an unintuitive one. By representing the pitch-velocity like this we introduce complexity for the model to learn i.e. as far as the model is concerned, \((m_n = 35, b_n = 5)\) and \((m_n = 35, b_n = 9)\) are unrelated since they have different unique tokens. However we know that they are in fact the same instrument. We remove this information from the model and ask it to learn this. We experimented with representing the velocity and pitch as separate tokens but found the results (subjective listening and quantitative evaluation of our model) to be better with the combined representation. We will revisit exactly why we think that is later.

Equation 2.6 concludes the velocity representation of our sequences.

\[
\text{seq} = [(pv_1, t_1), (pv_2, t_2), \ldots, (pv_N, t_N)] \text{ for } n \in [1..N] \tag{2.6}
\]

\(pv_n\) = Unique (pitch class, velocity bucket) token for \(n^{th}\) note

3. Time Representation

The time ordering of our current sequence can be deduced from the \(t_n\) values (the second dimension of the sequence elements). We want to reduce the number of dimensions at each element from two to one. To do this we insert special time tokens into the sequence to separate the pitch-velocity \((pv_n)\) tokens by tokens representing the time between them. After this step, the sequence ordering is integral to the interpretability of the sequence.

The transformation of the sequence in equation 2.6 is as follows

\[
\text{seq} = [pv_1, < t_2 - t_1 >, pv_2, < t_3 - t_2 >, \ldots, < t_N - t_{N-1} >, pv_N] \text{ for } n \in [1..N] \tag{2.7}
\]

\(< t_b - t_a >\) = Time tokens representing difference in time between notes \(b\) and \(a\)
To create the time tokens to fill the sequence in equation 2.7, the difference in time (in seconds) is computed between neighbouring pitches and converted to ticks. Ticks are a unit of time in MIDI representation that reflect the maximum resolution at which the MIDI recording software can detect notes. If the difference in time between two MIDI events is smaller than the length of a tick, they are recorded as occurring simultaneously, in our dataset the number of ticks per quarter is 480. Representing silence using ticks removes the implicit embedding of tempo from the sequence and is inspired by the successful application in a musical context using the Transformer-XL framework by Donahue et al in 32.

The number of ticks between two $pv_n$ events is designed to be as efficient, we want to reduce unnecessary complexity for the model to learn. As such there are 5 unique tick time tokens, found in Table 4.

| Time Token | Number of Ticks |
|------------|-----------------|
| 1          | 1               |
| 2          | 10              |
| 3          | 100             |
| 4          | 1000            |
| 5          | 10000           |

Table 4: Time Tick Tokens

Silences are filled with as few tick tokens as possible for the duration, for example a silence of 345 ticks is represented by $[3, 3, 3, 2, 2, 1, 1, 1, 1, 1]$ (3 × one hundred tokens, 4 × ten tokens and 5 × one tokens). Similarly a silence of 5003 ticks would be represented by $[4, 1, 1, 1]$. As such, any silence can be represented using just 5 time tokens.

There is a trade off here. By representing silence like this rather than a unique token for every time gap (i.e. token=1 for 1 tick, token=2 for 2 ticks etc...) we limit the amount of tokens the model has to learn for, and ensure that there are training examples of all tokens it is likely to find in unseen data at prediction time (5 with our current choice). However we could have represented the silence with just one token (for example token=1 for 1 tick) and just used that base unit token as many
times as necessary to fill the gap. The problem with this is now the complexity has been shifted from \textit{vertically} in the token space at a given time step to \textit{horizontally} across time. The model we use is powerful in the long-term but the extent of that power is still limited computationally, and these limits are tested most in the time dimension. The decision was chosen as a happy medium between providing the model with a sufficient amount of training examples for all tokens (as with the pitch class reduction) and packing as much information as possible into the space we have.

These time token sequences fill the $< t_b - t_a >$ gaps in equation 2.7, resulting in a one dimensional sequence of tokens (pitch-velocity and time). Pitches that are hit in unison are represented by neighbouring $pv$ tokens without any time tokens in between.

All of our sequences are converted to this one-dimensional format and joined together into one long stream. Each sequence is divided in the stream by a special dividing token. This joining is relatively infrequent in the stream as a whole and does not skew the models learning of tokens we care about. This approach is used to separate documents in the paper presented with our transformer-xl model [29] and to separate musical sequences in [32].

2.3 Modelling

For our modelling we use a Transformer-XL architecture [29]. From the original paper...

\textit{Transformer-XL builds on the Transformer architecture by augmenting it with a recurrence mechanism. The recurrence mechanism enables Transformer-XL to use information beyond its training segment by learning how to incorporate recurrent state from previous segments. In contrast, the original Transformer is only able to alter its predictions based on the current training segment, hence the available system memory during training is a bottleneck to its ability to learn long-term dependencies.}
Chapter 2. Architecture

The inability to effectively learn and generate in the long-term was an issue pointed out by Gillick et al in their LSTM approach presented with the Groove dataset \[31\]. Given previous success on similar tasks using the Transformer-XL architecture \[32\], we decided to use this model.

2.3.1 Transformer-XL Model

For a corpus of tokens \(x = (x_1, ..., x_T)\) the Transformer-XL model learns the joint probability \(P(x)\), auto-regressively expressed as

\[
P(x) = \prod_t P(x_t | x_{<t})
\]

As with the the original Transformer model \[24\], the conditional probability is learnt by training an encoder on the context, \(x_{<t}\) to a fixed hidden state which is then multiplied by the existing token embeddings, returning logits. A softmax is applied to the logits to give a categorical probability distribution for the next token \[29\].

The XL model is specifically interested in encoding arbitrarily long contexts (input sequences of arbitrary lengths). Traditionally this is achieved by breaking the input sequence into training segments and training the model individually on each. This results in the largest possible dependency length being dictated by the segment size and inevitably (more often than not) contexts being split up (in the event of a segment boundary falling in the middle of one of our concatenated input sequences).

To address these two limitations, the XL model implements a segment level recurrence mechanism, where the hidden state learnt for each segment is cached and made available to the next segment. Figure 6 illustrates this.

Applying this mechanism to every two segments creates a recurrence that effectively spans the length of all segments. This is noted as contributing to a huge increase in dependency length over the original Transformer or previous RNN models (450% and 80% respectively) \[29\].
2.4 Sampling and Generation

We want to use our learnt model to generate new sequences. Equation 2.8 provides the foundation for both of our generation tasks. The model has no explicit musical knowledge, as such the desired output is specified in number of tokens. Given that these can be a mix of pitch and time tokens, it is impossible to specify to the model the length you require in musical terms (such as number of beats for example).

2.4.1 Task 1: Generation

The generation task is to create new sequences completely from scratch. To do this the model is primed with the special token used to delimit sub-sequences in our long one-dimensional training sequence (from Section 2.2).

As mentioned in the previous section, the current token (in this case the special delimiter) is encoded and multiplied by the existing token embeddings to produce a logit distribution over the next token. We sample from this distribution to select our next token, feed this back into the model to update the memory/add to context and repeat until a given generation length.

Adopted from [32], there are two user specified parameters that alter this process.

- **Top K** - Before sampling from the logit distribution we take the top K most probable tokens, isolate them and normalise there probability distribution to some to one, the final sampling is done from this distribution.
• **Temperature** - This parameter alters the extent to which we truly sample from the distribution or just take the most probable.

Playing with these two parameters gives varying results. A higher (or no) top k affords more *improvisation*, or encourages less likely generations. Temperature dictates to what extent the generated rhythms are consistent over time.

The output of the generation is a sequence identical (in format) to that introduced in 2.7. We de-tokenise this sequence by creating a MIDI with the pitch and a velocity randomly sampled from within the bucket corresponding to the $pv$ value.

### 2.4.2 Task 2: Continuation

Generation by continuation functions exactly the same as generation introduced in the previous sub-section except that before generating the model is *primed* with an existing input sequence. That is to say that an existing input sequence is passed to the model, updating the internal memory and context before any sampling is done.

Temperature and Top K are both parameters of Continuation. Another parameter specific to continuation is the *prime length*. This is how many tokens from the priming sequence to pass to the model before asking it to generate. A higher value for prime length results in a much more stable output truer to the original form, however this comes and the cost of improvisation or exciting/interesting results.

### 2.5 Evaluation in Development

The framework used in this experiment is highly parameterized. The selection of these parameters is based on a mixture of subjective evaluation of the output (for data representation parameters) and perplexity of the predictions at training time (for model hyperparameters).
2.5. Evaluation in Development

2.5.1 Perplexity

Perplexity is a typical metric for evaluating auto-regressive language models offline, defined in equation 2.9 [33].

\[
PPL(X) = \exp\left(-\frac{1}{l} \sum_{i} \log_{\theta}(x_i | x_{<i})\right) \tag{2.9}
\]

Where \( \log_{\theta}(x_i | x_{<i}) \) is the log-likelihood of the \( i^{th} \) token primed on the antecedent context, \( x_{<i} \). This can be conceptualized as the average log of probabilities of a given token, given the previous context, across all tokens in a sequence. In our case, this sequence is the one-dimensional stream generated from the test/valid stratification of the data. A high probability is an indicator of a performant model and hence so is a low perplexity (owing to the negative sign and logs).

Model training is stopped when valid perplexity ceases to decrease. The model from this point is taken forward to be used for experimentation.
Chapter 3

Evaluation, Experimentation and Analysis

We subject our best model/data representation to a number of tests/analysis’ to understand to what extent we have learnt a musically valuable representation of rhythm.

500 individual rhythms of varying length and genre were generated for analysis. All generations were created via the generation or continuation methodologies (outlined in Sections 2.4.1 and 2.4.2 respectively).

As previously mentioned, similar tasks to what we attempt in this work have been achieved on the same dataset by Gillick et al in their Learning to Groove paper presented with the dataset [31]. Where possible we benchmark our results to those found there.

3.1 Offline Evaluation - Model Perplexity

Perplexity (Section 2.5.1) was used as our guiding metric for offline evaluation during training and model tuning. This serves as an informative and easy to calculate measure of how well our model predicts the data in our training set and as such was used to justify decision making in development (e.g in the representation
of the data and hyperparameters of the model).

The perplexity of our final model on our test dataset is 1.552\textsuperscript{1} Differences in the reporting of model quality at training time in \cite{31} makes it impossible to compare directly with Gillick. Though DuBreuil notes a perplexity of around 1.741\textsuperscript{2} on the valid set (which we can reasonably expect to be an absolute lower bound to his unreported test evaluation) in his book, *Music Generation with Magenta* \cite{34} using the Drums RNN model provided pre-packaged as part of the Magenta Project \cite{35}. Perplexity is useful because it is easy to obtain and tells us something directly about how our model understands the problem. It is however widely accepted not to necessarily be strongly correlated with human perception of musicality. In fact almost all key publications in the field of creative machine generation use evaluation methods encoded with a human preconception of what we consider valuable. In the specific case of musical creativity this often involves listening tests to measure, for example, musicality \cite{27}, pleasantness \cite{36} or - as in our benchmark paper, *Learning to Groove* - the ability to pass as human to the listener \cite{31}. It is the latter that we are concerned with here.

### 3.2 Online Evaluation - Listening Experiments

#### 3.2.1 Test Samples

Listening experiments were carried out using 500 samples generated by the continuation and generation methods. These samples were not cherry-picked and every generation was made available for the test.

The model and generation methods are parameterised so as to generate a token sequence of pre-specified length (more detail in Section 2.4\textsuperscript{3}) . As such, sequences of 3000 tokens were generated and the first 8 bars were extracted manually. This manipulation alongside the alignment of the first beat to coincide with time=0 is

\textsuperscript{1}Remembering from Section 2.5.1 that the lower the perplexity, the better the model

\textsuperscript{2}Important to note that this metric is reported as a log loss of approx 0.8 and that perplexity is computed as $2^{0.8} = 1.741$
the only human interference with the samples. We will revisit the full length of the sequences for velocity analysis in Section 4.3.

Given the imbalance in genre in the dataset and finite sampling for our test, some of the less common genres were not present. Table 5 displays the genres included in the test and their relative proportions. Of course, this is only relevant to those samples created by the generation method.

| Genre       | Rock | Reggae | Latin | Afrobeat | Soul | Punk | Dance | HipHop | Funk |
|-------------|------|--------|-------|----------|------|------|-------|--------|------|
| Prop        | 0.32 | 0.08   | 0.14  | 0.11     | 0.05 | 0.06 | 0.05  | 0.15   | 0.05 |

Table 5: Proportion of Samples with Genre in Listening Test

3.2.2 Experiment Setup

The experiment was facilitated by the Amazon Mechanical Turk platform on which workers were asked to listen to two 8-bar samples - one from our generated dataset of 500 and one from our original Groove MIDI dataset. Workers were aware that one of the two samples was generated by a machine, and one by a human. They were asked to elect which one they believed was generated by a human, they also had the option of answering with "Not sure".

Inspired by [32], to ensure that we only count responses where the worker genuinely listened to both samples we included 4 instances in which randomly generated noise samples replaced our machine-generated ones. Responses from workers who failed to identify the correct sample in any one of these 4 instances were removed from the test.

In total, 640 individual listening tests were carried out. After filtering out the responses of workers who failed the random noise test, we were left with 548 responses for analysis.

3.2.3 Results

Figure 7 illustrates our results. Standard error is calculated using a binomial proportion confidence interval of 95%.
3.2. Online Evaluation - Listening Experiments

Accuracy in Identifying Human Generated Rhythm

[7a] and [7c] show the accuracy of experiment participants ability to identify which of the pairs of samples they were presented with was human-generated - an accuracy of 60% indicates that 60% of the time, our model was not able to convince a human listener that it itself is human and hence a lower value in these charts supports a more performant model.

These two charts are split across the little metadata we had about the samples, genre and generation type. It is important to note that there is no ground truth genre annotation for the samples generated by the generation method (i.e. completely sampled from the model) and as such our sample size for experiments tagged with this information is roughly halved - hence the larger error.

Finally and most importantly, we have included in Figure [7a] the results of an almost identical listening experiment presented in Learning to Groove from Gillick et al [31]. In which their generations were put to listeners in a blind test in an effort to determine their models ability to pass as human. Though none of the three methods presented by Gillick match exactly the work achieved in this paper, we believe that the tasks are sufficiently similar enough to merit comparison - both papers are concerned with learning a model of expressive performance on the Groove MIDI dataset, both with the intention of using this model to predict and generate new and bespoke rhythms to equal or better human performance in musical creativity tasks.

Sureness in Annotation

In total, 77 out of 548 (14.1%) tests resulted in the listener not being able to identify which of the two samples was human (answering with "Not Sure"). Figure [7b] shows this proportion over all tests and for each of our generation methods separately.
Chapter 3. Evaluation, Experimentation and Analysis

(a) Overall Accuracy

(b) Sureness

(c) Genre Accuracy

Figure 7: Listening Test Results

3.3 Offline Analysis - Velocity Distribution

It is interesting to observe how the distribution of velocity across measures compares between our original dataset and our generated samples. This is naturally best achieved aurally (and for that reason we encourage the reader to spend time listening to the samples produced and provided alongside this document) however we also see value in visualising and qualifying them here.

3.3.1 Velocity Analysis Method

We work from the same pool of samples introduced in Section 4.2.1 however this time we do not reduce them to 8-bars in length (though we do maintain the alignment to the bar).

At each 16th time-step, the velocity of any existing pitches are summed so as to
produce one (aggregated) velocity value at 16th time-intervals across the entire sample. Finally for each 16th step in the bar, these velocities are averaged, resulting in 16 values each indicating the average summed velocity at each 16th step in the bars across the piece.

An example from one sample in the Afrobeat genre can be found in Figure 8, a typical bar from the MIDI is displayed alongside (where depth of colour illustrates velocity) and the audio is available at the link in the caption. The dashed line through the middle of the coloured section of the velocity plot is the average summed velocity at that point in the bar. The upper solid line (and border of the coloured section) represents one standard deviation above the average summed velocity at that point in the bar - conversely, the lower solid line (and border) is one standard deviation below. We encourage the reader to see whether they can match up the areas of higher velocity density in the MIDI with peaks in the velocity graph.

The visual continuity of the graphs has been chosen for aesthetic reasons; to more easily visualise how variance (thickness of coloured area) compares between neighbouring steps and to match intuition around continuity of music and how intensity of sound naturally decays over time. This is as oppose to having been fueled by actual numbers in the raw dataset - the values which are being plotted in the velocity distributions exist discretely at 1/16th timesteps and the joining of these discrete values with a continuous curve is achieved by fitting a spline of second-order polynomials.

The nature of the music as heard is one of a steady dance beat with a pulsing kick on the 0 and half step with rides of uniform intensity on the 1/8ths (present throughout the entire sample hence consistent peaks at 1/8ths). Each bar is carried by two quadruplets of snares, each of reducing intensity; the first is intertwined at odd time steps (3,5,7,9)/16ths creating the characteristic syncopation associated with afrobeat; the second begins hard at 12/16ths (notice the higher peak) and continues for 4 consecutive 1/16ths until 15/16ths. The syncopation of the first quadruplet and subsequent snare hits populate otherwise vacant areas in our plot/sequence (this

\[\text{3}\text{Remember that the velocity plot is an average across all measures. The example MIDI displayed is the first bar of many in the sample and serves to be indicative of the rhythm.}\]
accounts for the minimas at 1/8ths where generally the snare is the only instrument
clocked). Most bars in the sample begin (0/16) with high intensity (usually three
ingstrument), evident in the large peak on the chart. This resolves the tension built
up from the snares towards the end of the last bar and what generally follows is a
momentary pause in intensity before repeating.

This description is an attempt to aggregate and describe the essence of the music over
its measures but naturally each bar in the sample varies, this variance is captured
in the thickness of the coloured area at a given time step. It is not expected that
the reader can deduce this textual interpretation of the rhythm from the graph
alone. What is intended here is that the reader can visualise quickly some key
aspects of the music. For example in our afrobeat sample, certain key characteristics
are present; the syncopation is evident in the wavey, up-and-down nature of the
curve; the intensity peaks are on the 0 and 12 (typical in many west-african and
latin musical traditions); and the large(ish) variance in intensity over the whole bar
(indicative of variation across the piece). These aspects are different across different
styles and traditions. The idea of these visualisations is to provide a method of
identifying whether our model has captured or preserved these characteristics.
3.3.2 Continuations

Regarding our samples created by continuation, our goal is to (1) understand to what extent the model has learnt the input rhythm and maintained the structure effectively and (2) understand to what extent the model has added its own flavour/interpretation to the input rhythms, creating new rhythms of its own.

We will exhibit three pairs of samples broadly indicative of the continuation dataset as a whole.

**Afrocuban**

Original - [https://ladylane.uk/orig_afrocuban_vel_paper/](https://ladylane.uk/orig_afrocuban_vel_paper/)
Continued - [https://ladylane.uk/cont_afrocuban_vel_paper/](https://ladylane.uk/cont_afrocuban_vel_paper/)

**Rock**

Original - [https://ladylane.uk/orig_rock_vel_paper/](https://ladylane.uk/orig_rock_vel_paper/)
Continued - [https://ladylane.uk/cont_rock_vel_paper/](https://ladylane.uk/cont_rock_vel_paper/)

**Latin**

Original - [https://ladylane.uk/orig_latin_vel/](https://ladylane.uk/orig_latin_vel/)
Continued - [https://ladylane.uk/cont_latin_vel/](https://ladylane.uk/cont_latin_vel/)

**Listening Analysis**

The continuations are quite good, that is to say, they are consistently relatively musically advanced; they play in (and keep) good time; accents are in the right places; they exhibit many interesting and varied syncopations; in most examples/genres there is an identifiable, long-term structure with both repetition and one-off surprises (over time intervals of 8 bars+). There are very few examples of continuation where the model lost some aspect of rhythmic musicality that would give it away as being machine made (for example losing time, missing a beat, unusual velocity progressions) - the same cannot be said for the samples produced by the generation method which we will talk about in the next section.

The reason for this is evidently the models ability to mimic the input pattern in
the long-term. The continuations, though musically impressive, do not differ much (if at all) from the samples which they succeed. The more complex rhythms like afrocuban or latin feature less "improvisation" in the continuation than more simple ones like dance, rock or punk. This becomes obvious when we look at the velocity distributions.

Comparing Velocity Distributions

Figures 9, 10, 11, 12, 13 and 14 visualise the original and continuation’s velocity distributions across the bar.

![Figure 9: Velocity Distribution of Afro-Cuban Rhythm from Training Set](image)

![Figure 10: Velocity Distribution of Continuation of Afro-Cuban Rhythm from Model](image)

It is obvious in the case of afrocuban (Figures 9 & 10) that little if any improvisation was achieved by the continuation algorithm. The model almost exactly reproduces the input. As mentioned, this is observed in many of the higher complexity rhythms where presumably the probability of returning a more improvisational pitch (i.e. one not present in the rhythm so far) is forced to essentially zero in the distribution
from which the next token is sampled - this intuitively makes sense since the input structure is a lot less common and the model has learnt less paths out of it. This trade off between originality and maintaining the original structure is controlled (to the extent that it can be) by the temperature parameter \(2.4\), an increase of which makes improvisation more likely (0.92 was used for this experiment).

For the rock samples (Figures 11 & 12) we observe again that the original structure is preserved well although here we see a little more improvisation in the variance of velocities. One thing noted across all samples is the models tendency to vary the velocity of the output a lot more than what we observe in the input. Where for a lot of samples the human had played quite tight (see the narrower parts of the curve in Figure 11), the model had been a lot less predictable (see identical points in 12). This we can attribute to how velocity is represented in our token sequence - the continuous scale of velocities is bucketed into 4 evenly spaced buckets. When generating, our algorithm samples randomly from a continuous uniform distribution in the bucket specified by the model. This affords the model much more bandwidth.
than observed in the continuous, human made velocity patterns, for example, where
the original drummer may have altered velocities between neighbouring beats by a
couple of units, if this alteration traverses a velocity bucket boundary, the difference
in velocities in the generation has potential to be up to 2 bucket widths apart.

For the latin rhythms (Figures 13 & 14) we observe a fair amount of improvisation
in the audio of the sample. Though we notice from the velocity plots that the overall
accent placement remains in tact, one big change comes on the second 2/16th were
previously silence prevailed. Again we notice a big change in variance in velocity
across the sample. This latin continuation is a rare (though not too rare) example
of where the essence of the original input is maintained but a reasonable degree of
improvisation is added to it.
3.3.3 Generations

With no genre annotations our goal here is to see whether our model has managed to produce rhythms with distributions similar to what we have observed in our human samples. As with all our examples, these distributions are best evaluated aurally, however we have selected three trios of generations to visualise and discuss here, chosen to be representative of roughly three categories identified across all generations.

The Good

Good Sample 1 - [https://ladylane.uk/good94/](https://ladylane.uk/good94/)
Good Sample 2 - [https://ladylane.uk/good28/](https://ladylane.uk/good28/)
Good Sample 3 - [https://ladylane.uk/good18/](https://ladylane.uk/good18/)

The Bad

Bad Sample 1 - [https://ladylane.uk/bad16/](https://ladylane.uk/bad16/)
Bad Sample 2 - [https://ladylane.uk/bad25/](https://ladylane.uk/bad25/)
Bad Sample 3 - [https://ladylane.uk/bad29/](https://ladylane.uk/bad29/)

The Ugly

Ugly Sample 1 - [https://ladylane.uk/ugly31/](https://ladylane.uk/ugly31/)
Ugly Sample 2 - [https://ladylane.uk/ugly20/](https://ladylane.uk/ugly20/)
Ugly Sample 3 - [https://ladylane.uk/ugly38/](https://ladylane.uk/ugly38/)

Listening Analysis

Given that the three clusters of generations (good, bad and ugly) are very loosely defined and that classifying samples into these groups is not at all clear-cut, it is difficult to say exactly what proportion of our generated samples fall into each. The Good Samples however do make up a majority proportion of all those that were generated, together with the Ugly Samples, they make up at least 75% of all generations.

The Good are defined as such because by our own judgement they are musically
decent, consistent (they keep and remain in time), occasionally exciting, maintain long-term structure (over 8 or 16-bar loops) and could reasonably pass as human generated. However there isn’t much variation in style across the samples, largely they tend to be variations around rock, soul or dance beats with more complex rhythmic patterns such as those found in latin or afrocuban not appearing to any measurable degree. This last point is unsurprising given the distribution across genres in Groove Dataset we trained with (see Table 2).

The Bad are exactly that, they are poorly timed, the velocity is monotonous, accents are incorrectly placed and musically they warrant little merit. Often these samples were found to be the result of the model getting stuck in a bad loop, this then feeds back into the model by updating the internal state for the next generation creating a poor structure in the long-term.

The Ugly are interesting and make up a non-negligible part of our generations. These are samples deemed to exhibit some degree of musicality but a trained ear could identify that they were not played by musicians. For example they keep bad time, or the periodicity of some of the sub-rhythms do not match up with what is customary/expected/consonant. It is possible that these samples could fool a listener with no interest/experience in music into believing it was made by a human, or feasibly that it was played by an inexperienced drummer - an important point to bear in mind given that the listeners in our listening tests did not necessarily have experience in music.

Velocity Distribution

Figures 15, 16, 17 show the three velocity distributions for Good, Bad and Ugly Samples 1. The aspects of the graphs from the previous section that we earlier described as musically valuable are evident in our Good Sample 1 plot (fig. 15); even oscillation indicating accents and syncopation in the expected place, velocity peaks at 1/8th notes in line with the more rock rhythm we can hear and less variance in certain parts of the bar and more in others.

However in our bad and ugly plots (fig. 16 & 17) - where the machine did not play in
3.3. Offline Analysis - Velocity Distribution

time and velocities were more monotonous - we see a much more even distribution of velocities across the bar with large variance. It is quite obvious from the plots which of three is more musical and which is closer to random noise. It would be interesting in future work to incorporate this velocity information into the model generation, so as to prevent or dissuade the model from pursuing undesirable forms at generation time.

Figure 15: Velocity Distribution of Generation, Good Sample 1

Figure 16: Velocity Distribution of Generation, Bad Sample 1

Figure 17: Velocity Distribution of Generation, Ugly Sample 1
Chapter 4

Discussion

We hope to have demonstrated to a reasonable degree the value in applying Transformer neural networks to the task of automated rhythm generation for the purposes of appreciable musical output. We have presented both quantitative and qualitative examples of success and musicality in three key areas; quality of our model as a probabilistic representation of musical sequence, ability of our model to compete with professional human drummers and tendency of our model to output sequences with long term musicological features common in human drumming.

What follows are some interesting reflections worth touching on in conclusion.

Data Representation

The representation of the Groove MIDI Data set, specifically the unique choices taken for the tokenization, were a large contribution of this paper. Specifically of interest is how velocity is represented as combined with pitch. We propose that the perceived increase in performance as a result of this representation is due to how the generative sampling is done (section 2.4). With pitches represented individually, there exists only one token in the vocabulary that corresponds to a given instrument e.g. kick drum. Therefore it only takes one occurrence of a less likely pitch being sampled where a kick drum should be to throw off the sequence. Conversely with our representation of velocity-pitch combinations. There exists 4 of each instrument
in the dataset and hence it is a lot less likely the model would make a mistake (or sample of mu. This decision and insight was made subjectively by listening to the outputs.

The time representation is also unique and unseen in other works. As mentioned in section 2.2, this is chosen as a trade off between complexity in token space or complexity in time. We believe we reached a nice balance. All time tokens appear to a more than sufficient degree in our training data (every silence contains at least one of each). And yet total increase in sequence length as a result of this is minimal (though admittedly this is hard to relativize exactly). In any case one thing we can be sure of is the ability of the model to keep time. Removing explicit quantization time steps from the tokenization did not remove the models ability to remain in time for long periods (minute scale). This was unexpected and provides a decent foundation to build less quantized versions in future, where microtimings are facilitated completely by the maximum granularity provided by this compact tick representation.

Finally on the point of quantization, it was obvious after listening to some of the training examples that the 1/16th note quantization that a lot of the rhythmic essence of particular playing styles was lost at training time. Although these were not necessarily removed at prediction time (i.e. they can still feed into the model), this has no doubt removed from our model valuable data to learn from.

Quality of the Results

The generations are varied in quality and limited in genre. It has also not been proven that the model adds any significant layer of improvisation to the existing samples in the raw dataset. We argue that this is not a negative point and that reproducing input is impressive since it demonstrates an ability to learn in the long-term, something identified as difficult or expensive in previous algorithms (generally samples were consistent over periods of minutes).

The genre distribution of our output samples reflects the distribution in our raw dataset, this is expected albeit slightly disappointing (as some of the more rhyth-
mically interesting genres were less common). A fine-tuning technique such as that proposed in [32] could aid in controlling these distributions. For example if we were to train on a larger, more general dataset in future and then fine-tune our model on specific genres to produce models that were *experts* in specific genres. In any case our generations in the genres were subjectively and analytically comparable to the samples of the same genre in the original dataset.

Ultimately, upon listening to the samples individually and reviewing the results of the listening experiment, it is not presumptuous to assume that, on balance, the ability of our model to generate is decent. On the time scale observed, our generations out perform any state of the art we have seen to date for this type of task (admittedly there hasn’t been many).

**Model Parameters**

The selection of model and generation parameters have a huge impact on quality and character of results. Some observations include...

A lower memory length in the generations from scratch helped avoid the model getting stuck in *bad loops* (ie musically undesirable loops). This is presumably because the model doesn’t feedback into itself as much as with a longer memory length and hence doesn’t *internalize* it’s bad learnings.

Top K is to be tuned relative to the number of tokens and dictates to some extent how much improvisation the model is allowed to do. Temperature also balanced this trade-off and was useful in defining the models ability to find its way out of undesirable loops. As noted in [32], lowering the temperature prevented the generations from getting stuck in loops (both desirable and undesirable), though lowering it to a certain degree sacrificed musical quality, this is the trade off here.

A high enough prime length in continuation ensured a reliable reproduction of the input, but this comes at the cost of less experimentation. This balance was found subjectively on a handful of samples and applied to the whole dataset. There could perhaps be more effective ways of doing this on a per sample basis based on the
evaluation methods in the previous section.

**Listening Tests**

Finally and most importantly we reflect on the results of the listening tests. Given the statistical uncertainty presented on the results in Figure 7 it is impossible to conclude that the model performed better for a specific genre or task. However we can conclude that our model was consistently able to convince listeners that it was human and that this feat has not necessarily been completed on all generation tasks on this dataset to date. Listening to the generated samples corroborates these results in both the short and long term. An achievement that we present for the first time in this domain.
Chapter 5

Conclusions and Future Work

We have presented a successful attempt at the statistical modelling of musical rhythm for the purposes of automated rhythm generation in the long term (minute scales) using transformer neural networks. We present for the first time in this domain generations of musical quality comparable to human drummers both in musical character and how they are perceived. And in doing so we hope to have offered an exciting basis for the future development of percussion specific automated generation techniques.

There is always more work to do in such a quickly developing field and we hope to have outlined some of the points of improvement in this work throughout the document but would like to draw attention now to some more important lines of investigation for the near future.

- Pre-training on a larger dataset and fine-tuning on groove genres for genre specific generations/better model quality. Given how transformers usually find their greatest success on very large datasets, this is almost definitely going to improve the quality of the model. The Lakh MIDI dataset would be a good candidate for this. [37]

- A deeper investigation into micro-timings and how the model captures/generates these.
• Incorporating our evaluation methods at generation-time methods to ensure high quality in output.

• Implementing the ability to specify how many bars/beats of a generation is required before hand.

For now though we thank the reader for their attention and welcome any future insight, improvement or feedback on the methods presented in this document.
List of Figures

1 An unrolled RNN [9] ............................................. 3
2 Repeating module of standard RNN [9] .......................... 4
3 Repeating module of standard LSTM [9] ........................ 5
4 Attention Matrix for Translation Task [21] ....................... 7
5 Schematic of Basic Transformer Module [21] ..................... 8
6 Recurrence Mechanism of XL model with Segment Length = 4 [29] . 19
7 Listening Test Results ............................................. 26
8 Velocity Distribution of Afrobeat from Training Set with Aligned MIDI From 1st Bar [https://ladylane.uk/cont_afrobeat_vel_plot/] ............................................. 28
9 Velocity Distribution of Afro-Cuban Rhythm from Training Set ............................................. 30
10 Velocity Distribution of Continuation of Afro-Cuban Rhythm from Model ............................................. 30
11 Velocity Distribution of Rock Rhythm from Training Set ............................................. 31
12 Velocity Distribution of Continuation of Rock Rhythm from Model ............................................. 31
13 Velocity Distribution of Latin Rhythm from Training Set ............................................. 32
14 Velocity Distribution of Continuation of Latin Rhythm from Model ............................................. 32
15 Velocity Distribution of Generation, Good Sample 1 ............................................. 35
16 Velocity Distribution of Generation, Bad Sample 1 ............................................. 35
17 Velocity Distribution of Generation, Ugly Sample 1 ............................................. 35
## List of Tables

| Table | Description                                           | Page |
|-------|-------------------------------------------------------|------|
| 1     | Train, Test and Validation Splits of GMD              | 11   |
| 2     | Genre Distribution of GMD                             | 11   |
| 3     | Pitch Mappings of Our Dataset                         | 13   |
| 4     | Time Tick Tokens                                      | 16   |
| 5     | Proportion of Samples with Genre in Listening Test   | 24   |


Bibliography

[1] Jacob, B. L. Algorithmic composition as a model of creativity. *Organised Sound* 1, 157–165 (1996).

[2] Alpern, A. Techniques for algorithmic composition of music (1995).

[3] Maurer, J. A. A brief history of algorithmic composition. [https://ccrma.stanford.edu/~blackrse/algorithm.html](https://ccrma.stanford.edu/~blackrse/algorithm.html). Accessed: 2020-02-10.

[4] Xenakis, I. *Formalized music: thought and mathematics in composition / Iannis Xenakis* (Pendragon Press Stuyvesant, NY, 1992), rev. ed. edn.

[5] Anderson, C., Eigenfeldt, A. & Pasquier, P. The generative electronic dance music algorithmic system (gedmas) 5–8 (2013).

[6] Ames, C. Automated composition in retrospect 1956-1986 (1987).

[7] Shottstaedt, W. *Automatic Counterpoint*, 199–214 (MIT Press, Cambridge, MA, USA, 1989).

[8] Briot, J., Hadjerès, G. & Pachet, F. Deep learning techniques for music generation - A survey. *CoRR* abs/1709.01620 (2017). URL [http://arxiv.org/abs/1709.01620](http://arxiv.org/abs/1709.01620), 1709.01620.

[9] Understanding lstms. [https://huggingface.co/transformers/perplexity.html](https://huggingface.co/transformers/perplexity.html) [https://colah.github.io/posts/2015-08-Understanding-LSTMs/](https://colah.github.io/posts/2015-08-Understanding-LSTMs/)

[10] Todd, P. M. & Loy, G. A connectionist approach to algorithmic composition. In *Computer Music Journal*, vol. 13, 27–43 (1989).
[11] Boulanger-Lewandowski, N., Bengio, Y. & Vincent, P. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. Proceedings of the 29th International Conference on Machine Learning, ICML 2012 2 (2012).

[12] Chung, J., Gülçehre, Ç., Cho, K. & Bengio, Y. Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR abs/1412.3555 (2014). URL http://arxiv.org/abs/1412.3555 1412.3555.

[13] Hochreiter, S. & Schmidhuber, J. Long short-term memory. Neural Computing 9, 1735–1780 (1997). URL https://doi.org/10.1162/neco.1997.9.8.1735.

[14] Eck, D. & Schmidhuber, J. Finding temporal structure in music: Blues improvisation with lstm recurrent networks. vol. 12, 747 – 756 (2002).

[15] Choi, K., Fazekas, G. & Sandler, M. B. Text-based LSTM networks for automatic music composition. CoRR abs/1604.05358 (2016). URL http://arxiv.org/abs/1604.05358 1604.05358.

[16] Johnson, D. Generating polyphonic music using tied parallel networks. In International Conference on Evolutionary and Biologically Inspired Music and Art, 128–143 (2017).

[17] Mao, H. H., Shin, T. & Cottrell, G. W. Deepj: Style-specific music generation. CoRR abs/1801.00887 (2018). URL http://arxiv.org/abs/1801.00887 1801.00887.

[18] Simon, I. Performance rnn: Generating music with expressive timing and dynamics. https://magenta.tensorflow.org/performance-rnn Accessed: 2020-02-10.

[19] Makris, D., Kaliakatsos-Papakostas, M. A., Karydis, I. & Kermanidis, K. Combining lstm and feed forward neural networks for conditional rhythm composition. In EANN (2017).
[20] Hutchings, P. Talking drums: Generating drum grooves with neural networks. *CoRR* abs/1706.09558 (2017). URL http://arxiv.org/abs/1706.09558.

[21] Bahdanau, D., Cho, K. & Bengio, Y. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473 (2014).

[22] Raffel, C. & Ellis, D. P. W. Feed-forward networks with attention can solve some long-term memory problems. *CoRR* abs/1512.08756 (2015). URL http://arxiv.org/abs/1512.08756.

[23] Bahdanau, D., Chorowski, J., Serdyuk, D., Brakel, P. & Bengio, Y. End-to-end attention-based large vocabulary speech recognition. *CoRR* abs/1508.04395 (2015). URL http://arxiv.org/abs/1508.04395.

[24] Vaswani, A. *et al.* Attention is all you need. *CoRR* abs/1706.03762 (2017). URL http://arxiv.org/abs/1706.03762.

[25] Uszkoreit, J. Transformer: A novel neural network architecture for language understanding. https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html Accessed: 2020-02-10.

[26] Devlin, J., Chang, M., Lee, K. & Toutanova, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR* abs/1810.04805 (2018). URL http://arxiv.org/abs/1810.04805.

[27] Huang, C. A. *et al.* An improved relative self-attention mechanism for transformer with application to music generation. *CoRR* abs/1809.04281 (2018). URL http://arxiv.org/abs/1809.04281.

[28] Payne, C. Musenet. https://openai.com/blog/musenet/ Accessed: 2020-02-10.

[29] Dai, Z. *et al.* Transformer-xl: Attentive language models beyond a fixed-length context. *CoRR* abs/1901.02860 (2019). URL http://arxiv.org/abs/1901.02860.
[30] Groove midi dataset. [https://magenta.tensorflow.org/datasets/groove](https://magenta.tensorflow.org/datasets/groove). Accessed: 2020-02-10.

[31] Gillick, J., Roberts, A., Engel, J. H., Eck, D. & Bamman, D. Learning to groove with inverse sequence transformations. *CoRR abs/1905.06118* (2019). URL [http://arxiv.org/abs/1905.06118](http://arxiv.org/abs/1905.06118).

[32] Donahue, C., Mao, H. H., Li, Y. E., Cottrell, G. W. & McAuley, J. J. Lakhnes: Improving multi-instrumental music generation with cross-domain pre-training. *CoRR abs/1907.04868* (2019). URL [http://arxiv.org/abs/1907.04868](http://arxiv.org/abs/1907.04868).

[33] Hugging face. [https://huggingface.co/transformers/perplexity.html](https://huggingface.co/transformers/perplexity.html). Accessed: 2020-08-30.

[34] DuBreuil, A. *Hands-On Music Generation with Magenta* (Packt, 2020).

[35] Magenta. [https://magenta.tensorflow.org/](https://magenta.tensorflow.org/). Accessed: 2020-08-23.

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

[37] Thierry Bertin-Mahieux, B. W., Daniel P. W. Ellis & Lamere., P. The million song dataset. In *International Society for Music Information Retrieval Conference*, vol. 12, 591–596 (2011).