Concurrent generation of melody and lyrics by recurrent neural networks

PIETRO BOLCATO

Master in Computer Science
Date: June 15, 2020
Supervisor: Bob L. T. Sturm
Examiner: Robert Lagerström
School of Electrical Engineering and Computer Science
Host company: Doremir
Swedish title: Samtidig generation melodi och texter av recurrent neural networks

Abstract

This work proposes a conditioned recurrent neural network architecture for concurrent melody and lyrics generation. This is in contrast to methods that first generate music and then lyrics, or vice versa. The system is trained to first sample a pitch from a distribution, then sample a duration conditioned on the sampled pitch, and finally sample a syllable conditioned on the sampled pitch and duration. The evaluation metrics show the trained system generates music and text sequences that exhibit some sensible musical and linguistic properties, and as further evaluation, it was applied in a human-AI collaboration for the generation of a song for the VPRO AI Song Contest. This highlighted the limitations of the system: it can be a useful tool to augment the creative process of musicians, but it can not replace them. Finally, a shorter version of this dissertation has been submitted as a paper for the ISMIR 2020 conference, and it is shown in appendix B.

Sammanfattning

Detta arbete föreslår en konditionerad återkommande neurala nätverksarkitektur för samtidig melodi och textgenerering. Detta i motsats till metoder som först genererar musik och sedan texter, eller vice versa. Systemet tränas för att först sampla en tonhöjd från en fördelning, sedan prova en varaktighet konditionerad på den samplade tonhöjden, och slutligen prova en stavning som är betingad av den samplade tonhöjden och varaktigheten. Utvärderingsstatistiken visar att det tränade systemet genererar musik- och textsekvenser som uppvisar några känsliga musikaliska och språkliga egenskaper, och som en ytterligare utvärdering användes det i ett mänskligt-AI-samarbete för att generera en låt för VPRO AI Song Contest. Detta markerade systemets begränsningar: det kan vara ett användbart verktyg för att öka musikernas kreativa process, men det kan inte ersätta dem. Slutligen har en kortare version av denna avhandling lämnats in som en uppsats för ISMIR 2020-konferensen och den visas i bilaga B.

Contents

Chapter 1

Introduction

The generation of a melody with lyrics is a challenging problem in the field of artificial intelligence. Melody generation involves deciding which pitch is played at a given time and for how long so that the composition follows common musical rules about rhythm and structure. Likewise, lyrics have to respect grammatical structures and in the case of syllable generation, consecutive syllables must form sensible words. Furthermore, lyrics and melody have to work together to create a convincing whole.

Previous works either considered the generation of melody and lyrics as two separated processes [1–5] or the generation of the former conditioned on the latter, and vice versa [6, 7]. This dissertation proposes modeling melody and lyrics concurrently as an extension of the conditional recurrent neural network architecture proposed in [2, 3]. There, the authors jointly model pitches and durations by first generating a duration and then a pitch conditioned on that duration. Here, I adopt a similar architecture and I add modeling of syllables. Thus, the network first generates a pitch, then conditioned on that pitch generates the corresponding note duration, and finally conditioned on the pitch and duration, generates the syllable. This process is repeated a predefined number of times. To the best of my knowledge, this is the first attempt on such a concurrent generation of notes and lyrics.

I adopt the dataset used in the work of Yu and Canales [7] and I represent the input data as sequences of pitches, durations and syllables triplets, with pitch and duration expressed as discrete MIDI [8] attributes. Once the network is trained, I evaluate its performance by comparing several evaluation metrics computed both on the training set and on the sequences generated by the network.

There are many reasons why investigating the automatic generation of mu-

sic is interesting [1–4, 6, 7, 9–16]. First, it is fascinating to explore if, and to what extent, an automated system can exhibit human-like properties and behaviours because as humans we often think that art and creativity are something reserved to our minds only. Second, from a commercial point of view, automated music production could fill a role in several different scenarios because playing AI-generated tunes would have the benefit to be cheaper and different at all times if compared to traditional human-composed songs. Finally, from a more artistic point of view, the automated generation could aid artists by becoming a part of their creative process through a human-AI collaboration [17, 18].

The collaboration between artists and AI is explored as well by DoReMIR, the research institute that hosted my internship. They are active in the area of automatic music notation, helping musicians to create scores while playing in real time. Furthermore, the company recently started a project aiming to generate a song given some user-defined input and the model proposed in this work could be integrated as part of it.

The dissertation is divided as follows: Chapter 2 presents an overview and a discussion of related works. Chapter 3 discusses the proposed system: its architecture, training, and evaluation metrics. Chapter 4 and 5 present and discuss the results and finally Chapter 6 concludes the dissertation and proposes possible future improvements.

The source code and the trained models are publicly available on Github at: `https://github.com/seicaratteri/aisongcontest`. Furthemore, several generated samples can be listened at: `http://seicaratteri.github.com/paper/samples.html`.

1.1 Research Question

The overall purpose of this work is to create, train and evaluate a neural network model able to generate melody and lyrics concurrently. Furthermore, the method should be applicable in a real case scenario, such as a human-AI collaboration. This implies the following research question: *How can melody and lyrics be modelled and generated concurrently by neural networks?* This aims to explore the feasibility of such a concurrent generation as well as evaluate it. Furthermore, the design of the network should fulfill the following conditions:

1. Model the generation of pitches, durations and syllables conditionally
2. Have no restriction on the length of the output sequence

3. Be conditionable to a manual input

Condition 1 ensures that the relation between pitch, duration and syllable attributes are effectively taken into account. Condition 2 and 3 ensure that the network can be steered and conditioned to a certain degree by the user, resulting in a system easier to use in a real case scenario.

1.2 Ethical, Societal and Sustainability Aspects

In general, it is difficult to assess the impact of an artificial intelligence system as one can never really know what its architecture and underlying ideas could be used for. One concern regards the idea that AI-enabled systems will replace workers across a wide range of industries but history, from the industrial revolution to computers today, teaches us that the impact of technologies is not pre-determined and it can be shaped through regulations and policies both on an local and global scale in order to produce a positive shift in job roles and create new categories of employments [19].

Nonetheless, as Holzapfel, Sturm, and Coeckelbergh [10] point out, music is a social phenomena and therefore music-related algorithms can have important social implications that must be considered. Take, for instance, the case of a music recommendation system: having more training data of a particular genre or artist can result in a recommendation bias which negatively affects artists belonging to under-represented data. In the specific case of the music generation system proposed in this dissertation, the dataset is a collection of Eurogenetic forms of music and therefore a intrinsic bias is present.

Furthermore, one could be concerned about legal issues such as copyright and authorship. Researcher and practitioners [9] analysed this controversial legal area. In many countries, there is no specific legal framework in place to deal with autonomously generated music and according to a large number of scholars, it might not be eligible for copyright protection. On the other hand, few other countries such as the UK, South Africa, Hong Kong, India, Ireland, and New Zealand "have envisaged protection for computed-generated works granted to the person by whom the arrangements necessary for the creation of the work have been undertaken [9]". This shows that the legal system all over the world is slowly changing and adapting to the rise of automatic music generation systems, but only time will tell how it will be precisely shaped.

Generally, it is a complex matter because of a variety of factors involved. First of all, it is not always easy to measure if, and to what extent, a work is

generated by artificial intelligence. Sturm et al. [9] discuss the possibility of enumerating the involvement of AI in the different stages of music production with a numerical score or a text that describe it precisely. This would give the benefit of complete transparency but at the same time it could be detrimental due to the fact that is not always easy to track the influence of AI in the hardware and software used by the artist during the creative process, and therefore could represent a notable barrier.

Furthermore, it is important to consider not only the autonomous music generation system in itself, but also the dataset used to train it. Generally, when the music used to train a system is protected by copyright, permission from the rightholders is required unless an exception applies [9]. A recent European directive[1] introduced an exception in the case of text and data mining for purposes of scientific research provided the researcher has lawful access to the work, resulting in a great aid to AI researchers.

Finally, it is important to consider plagiarism. There are no clear standards or conventions about what constitutes a sufficient alteration of existing materials to be considered novel, but efforts should be taken to ensure that generated material does not plagiarise existing music [9]. Indeed, an intelligent autonomous system should be able to create novel material rather than copying existing one.

To conclude, there are many legal frameworks that does not specifically address the issue of automated music generation. Given that there are no intrinsic unethical or malevolent purposes in the system proposed in this dissertation, and considering that is an extension of the already publicly available work of [2, 3], I decided to release the code and the trained models.

[1] Directive (EU) 2019/790 of the European Parliament and of the Council of 17 April 2019 on copyright and related rights in the Digital Single Market and amending Directives 96/9/EC and 2001/29/EC, OJ L 130, 17.5.2019, pp. 92–125.

Chapter 2

Background

The use of formal process to create music can be traced as far back in time as the ancient Greeks, where they built their musical systems using several mathematical properties and theoretical applications of numbers. For instance, the Greek mathematician and astronomer Ptolemy wrote about how musical notes could be translated into mathematical equations and vice versa in his book *Harmonics* [20]. Later in history, the canonic composition of the 15th century introduced a further layer of abstraction by giving a set of rules to determine additional voices starting from a single voice part. In more modern times, the advent of computers introduced new astonishing opportunities and the term *algorithmic composition* [21] was coined to define the process of using automated systems to generate music with minimal human intervention. Several techniques have been developed over the years, spanning from knowledge-based or rule-based methods to statistical models and deep learning approaches.

Traditional methods combine music knowledge representation with the addition of composition rules for algorithmic composition. For instance, Fernández and Vico [14] highlight several different systems that use Constraint Programming [22] to model music theories and compositions because the support for rule-like programming constructs makes the paradigm well suited to represent music theory models. At the same time, it has some intrinsic limitations: the rule engine should logically become nearly as complicated as the problem the system is trying to solve, and in the specific case of modelling music, this results in a very complex and opaque system where there is a high chance of creating conflicting or overlapping rules.

To overcome this issues, researchers explored statistical models such as Markov chains. These represent a way to model an abstract machine with dif-

ferent states that, based on a time sequence, will transition between its states with different probabilities. The work of Eigenfeldt and Pasquier [12], for example, is able to generate jazz chords progressions given as input a three dimensional vector which specifies the desired bass-line, harmonic complexity, and tension between chords transitions. Another work from Davismoon and Eccles [13] uses simulated annealing to combine constraints with Markov processes in order to generate melody and rhythm. However, the limitations of finite-order Markov chains for sequences are clear from the mathematics, which support their behavior in modeling music sequences: especially in the case of low-order chains the performance proved to be generally "poor, producing unmusical results" [14]. On the the other hand, high-order chains tend not to generalize enough to produce novel material and they are also computationally expensive to train.

Therefore, researchers and practitioners started to explore different methods such as artificial neural networks, and the last few years have seen a rise in their usage motivated by their performances in a wide variety of fields. In the case of music generation, this includes the generation of folk-music melodies using LSTM units that learn from music transcriptions expressed with a high-level vocabulary [1]. Another work [2] proposes a conditional recurrent neural network that jointly models pitches and durations, effectively taking into account the relation between a note duration and its pitch. A further variation [3] proposes a more general and normalized representation of music. Moreover, trying to extend the generation from monophonic to polyphonic music, another paper [4] trains a CNN-GAN on a piano roll representation and is able to generate five tracks including bass, drums, guitar, piano and strings.

Other works explore the relationship between melody and lyrics and they either try to generate the former conditioned to the latter, or vice versa. These approaches as well as the conditioned recurrent neural architecture proposed in [2, 3] are closely related to the topic of this dissertation and they constitute the basis on which this work is built. Therefore, I analyse them in greater detail in the following subsections.

2.1 Lyrics generation given melody

The work of Watanabe et al. [6] proposes a recurrent neural network language model able to generate lyrics from a given melody. To train the model, the authors first scraped a Japanese forum of musicians posting score and lyrics and collected a corpus of 54,181 lyrics. As a further refinement, they then created a collection of 1000 lyrics-melody pairs augmented with precise syllable-note

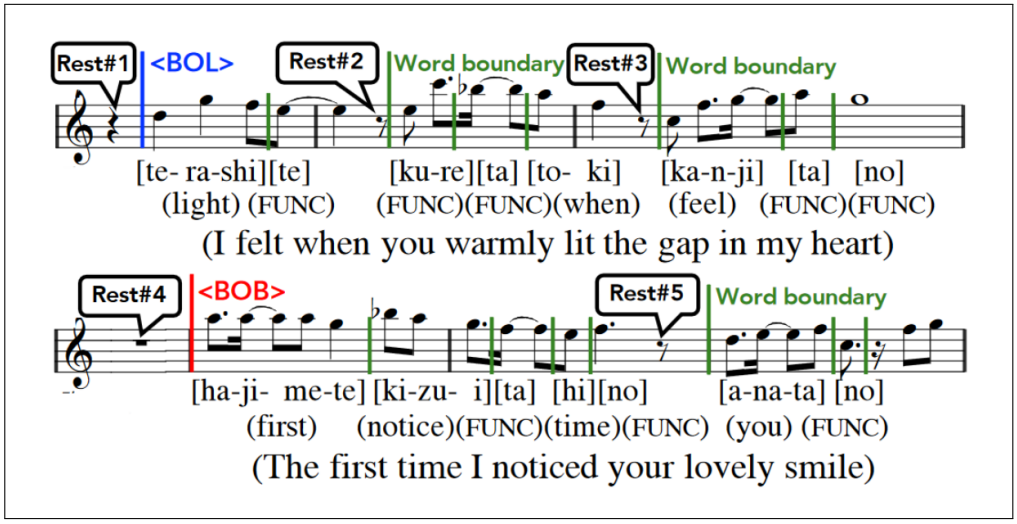

Figure 2.1: Example of the data collected by Watanabe et al. [6]. BOB indicates a block boundary and BOL a block of line. The figure is taken from the original publication.

alignments as well as word, sentence, and paragraph boundaries. Figure 2.1 shows an example of the data.

They conducted a statistical analysis of the data to get insights about the relation between music rest and lyrics. As result, they observed that the positions of lyrics segment boundaries are biased to music rest positions, and the probability of boundary occurrence depends on the duration of a rest, i.e., a shorter rest tends to be a word boundary and a longer rest tends to be a block boundary.

The authors trained a recurrent neural network language model able to generate lyrics and segments that follow the same distribution observed in the analysis described above. Given a melody consisting of notes and rests, the model outputs a one-to-one correspondence of words, boundary of lines or boundary of blocks. The generation of the word w_t is based on the previously generated word w_{t-1} as well as a context melody vector that takes into account a wider window of notes comprising of 10 time steps in the past and 10 time steps in the future with respect to the current one.

In order to train the network, the authors first split the dataset set in a proportion of 90% for training and 10% for validation. Furthermore, they preprocessed it by keeping only 20,000 of the most frequent words with syllable counts that are equal to or less than 10, while converting the others to a special symbol *unknown*. Moreover, to give a language prior to the model, the

authors pre-trained it on the large corpus of lyrics alone, effectively not taking into consideration the melody.

The evaluation of the results are based both on quantitative and qualitative metrics. Specifically, the quantitative metrics are: (1) perplexity, which measures the predictability of the words in the original lyrics; (2) F_1 measure of the boundary positions, which measures the consistency between the melody and boundaries in the generated lyrics. On the other hand, the qualitative evaluation include several criterias such as grammatical coherence and meaning of the lyrics which are evaluated by 50 Yahoo crowdsourcing workers.

The experimental results show that the model successfully captured the consistency between melody and boundaries of lyrics while maintaining word fluency and this was confirmed as well by the human judment collected via crowdsourcing. Examples of generated lyrics (in Japanese) can be found at: `https://github.com/KentoW/deep-lyrics-examples/tree/master/lyrics`.

2.2 Melody generation given lyrics

The work of Yu and Canales [7] aims to generate melody given lyrics. The lack of a large dataset of melody-lyrics paired data, especially on a syllable level, remains one of the biggest problem when tackling the generation of melody conditioned on lyrics or vice versa. Therefore, the first contribution of their work was to create a large-scale paired melody-lyrics dataset composed of 12,197 MIDI songs automatically aligned on a syllable level using the timestamp information of the MIDI tracks. Furthermore, the authors also trained a skip-gram [23] model to perform lyrics embedding both on a word and on a syllable level.

The proposed architecture, depicted in Figure 2.2, is a LSTM-GAN where the generator, given the lyrics embedding and a noise vector, generates a MIDI sequence. The discriminator, on the other hand, distinguishes the generated melody from real samples, and is trained by estimating the probability that a sample is from the real training dataset rather than the generator. In this way, generator and discriminator are trained in a adversarial manner until convergence.

The authors in order to evaluate the performances of the network adopted several evaluation metrics, explained below:

- Pitch distribution: a count of which pitches occur throughout the sequences

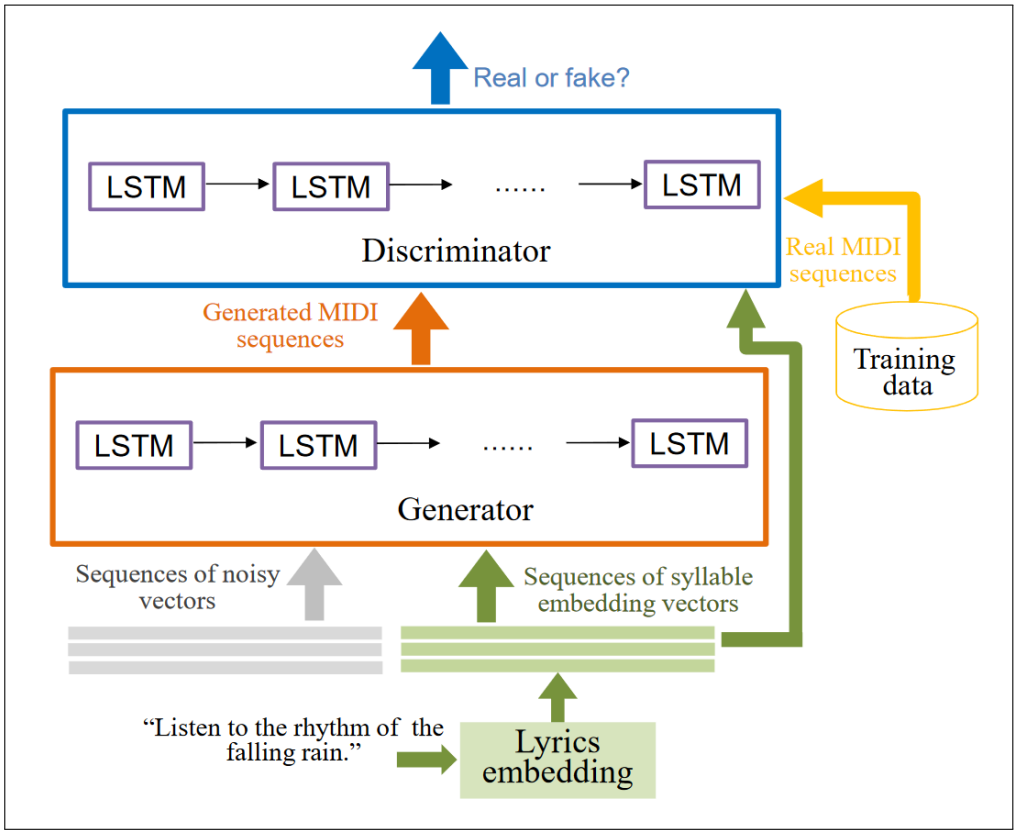

Figure 2.2: LSTM-GAN architecture used in the work of Yu and Canales [7]. The generator, given the lyrics embedding and a noise vector, generates a MIDI sequence. The discriminator, on the other hand, distinguishes the generated melody from real samples, and is trained by estimating the probability that a sample is from the real training dataset rather than the generator. The figure is taken from the original publication.

- Duration distribution: a count of which durations occur throughout the sequences
- Interval distribution: a count of which intervals occur throughout the sequences
- N-gram distribution: a count of how many MIDI numbers N-grams repetitions occur throughout the sequences
- MIDI number span: the difference between the highest pitch and the lowest one
- Number of unique MIDI numbers: a count of how many unique pitches are present in the sequence
- Song length: the sum of all the durations in a sequence

Most of the evaluated sample converge very closely to the ground truth value, calculated by computing the same numerical statistics on the training set. This indicates that the model successfully learned to approximate well the distribution of the considered attributes. Furthermore, experimental results proved that the conditioning on lyrics is successful. Examples of generated melodies can be found at: `https://github.com/yy1lab/Lyrics-Conditioned-Neural-Melody-Generation`.

2.3 Melody generation by Conditioned RNN

The work of Colombo, Seeholzer, and Gerstner [2] proposes a conditioned recurrent neural network architecture for the generation of monophonic melodies, trained on a corpus of melodies in Irish folk and Klezmer style.

As first preprocessing step, the authors converted the melodies from symbolic notation, i.e. *ABC* [24], to MIDI [8] notes sequences. Afterward, they built a integer vocabulary of pitches and durations, removing only rare events occurring less than 10 times in the whole dataset.

The proposed architecture, diagrammed in Figure 2.3, is composed of two separate recurrent neural network models. The idea is to explicitly divide pitch and duration, and to use two different networks to model them. These are trained jointly: the output of the duration network is given as input to the pitch network, effectively conditioning the generation of the latter to the former.

In order to assess the quality of the generated melodies, the authors designed a measure for creativity which indicates the novelty of a generated song

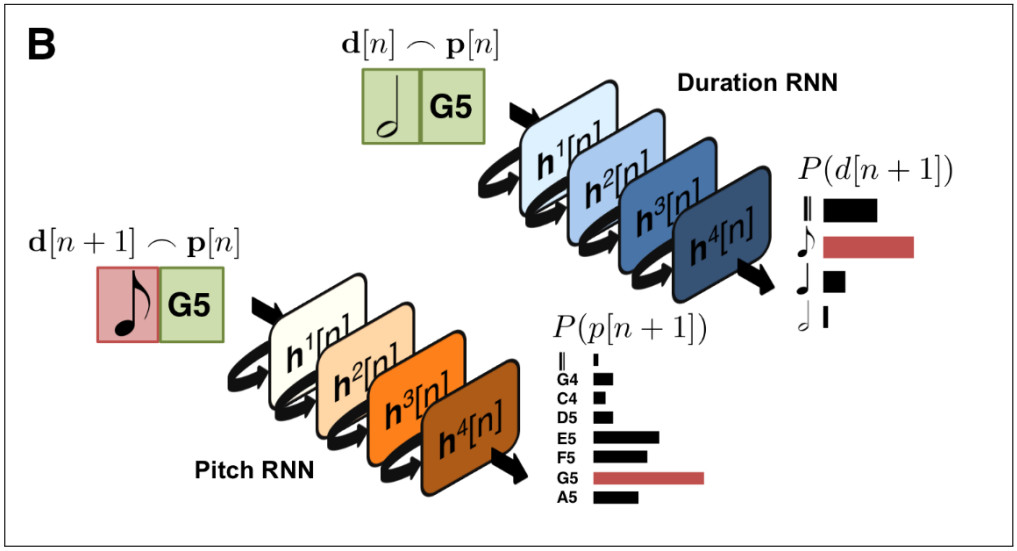

Figure 2.3: Conditioned recurrent neural network architecture used in the work of Colombo, Seeholzer, and Gerstner [2]. The system first generate a duration, and then a pitch conditioned on that duration. The figure is taken from the original publication.

s with respect to a corpus C and a motif size m. This is computed by calculating the fraction of the song's transitions which are not found in the musical corpus C, given only the last $m-1$ notes.

The novelty profiles calculated using this metric, show that the model is able to create melodies that are as novel as the validation set but, because the network needs to learn by example, they are naturally less novel than the training set. Furthermore, the authors analysed one melody generated for the Irish and Klezmer style and noticed coherence in terms of rhytmical patterns, melody structure and scale, showing that the network effectively learned what categorizes one style with respect to the other. Example of mp3 renderings of generated melodies can be found at: `https://www.dropbox.com/sh/51pttb63lu5v57n/AAA0MiXgIaHRQF9wbdS1zTkYa?dl=0`.

2.4 Lyrics generation

In the case of lyrics generation alone, the state of the art is reached by general purpose language models that can be trained or fine-tuned on a corpus of lyrics, and specifically the case of GPT-2 from Radford et al. [25] stand out for its outstanding results. The architecture adopted is a decoder-only transformer with

masked self attention in order to prevent token at the right of the currently considered one to get the priority over the left. The input is represented using byte pair encoding which is proven to perform well as it can capture words that share semantic meaning as well as words that are grammar-related only. Furthermore, the authors include character level, subword level and word level embeddings. The model was trained on a very large corpus of text scraped from reddit posts with minimum karma three, therefore ensuring that the text is grammatically correct and meaningful. Examples of generated texts can be found at: `https://openai.com/blog/better-language-models/`.

Chapter 3

Methods

This work focuses on generating melody and lyrics concurrently. The development of the proposed system did not proceed in a linear fashion, and a variety of decisions and experiments led to me exploring many different approaches, briefly described as follow.

The first architecture I tried was a LSTM-GAN similar to [7] with the addition of the concurrent generation of syllables. Unfortunately, it did not perform well due to the limitations of LSTM-GANs when it comes to the generation of finite tokens such as text. Indeed, if one uses a recurrent neural network, at every time step the next token is chosen by sampling from the output of the softmax function. This sampling operation is non-differentiable and therefore cannot be back-propagated from the generator to the discriminator.

To solve this issue, I tried the method proposed by Donahue and Rumshisky [26]. They use an autoencoder to learn a low-dimensional representation of sentences and train a MLP-GAN to generate its own vectors in this latent space, which are then decoded to realistic sentences. This method proved to work well for the generation of notes but failed in the case of syllables, and I thus discarded this approach.

I finally adopted a conditioned recurrent neural network architecture [2, 3], described in the following Section. Its architecture, training and evaluation are now explained in detail. All the code is written in Python, using Keras [27] with Tensorflow [28] as backend.

3.1 Neural network

The model this work uses is diagrammed in Fig. 3.1. It is a conditional recurrent neural network similar to that proposed in [2, 3]. The network takes

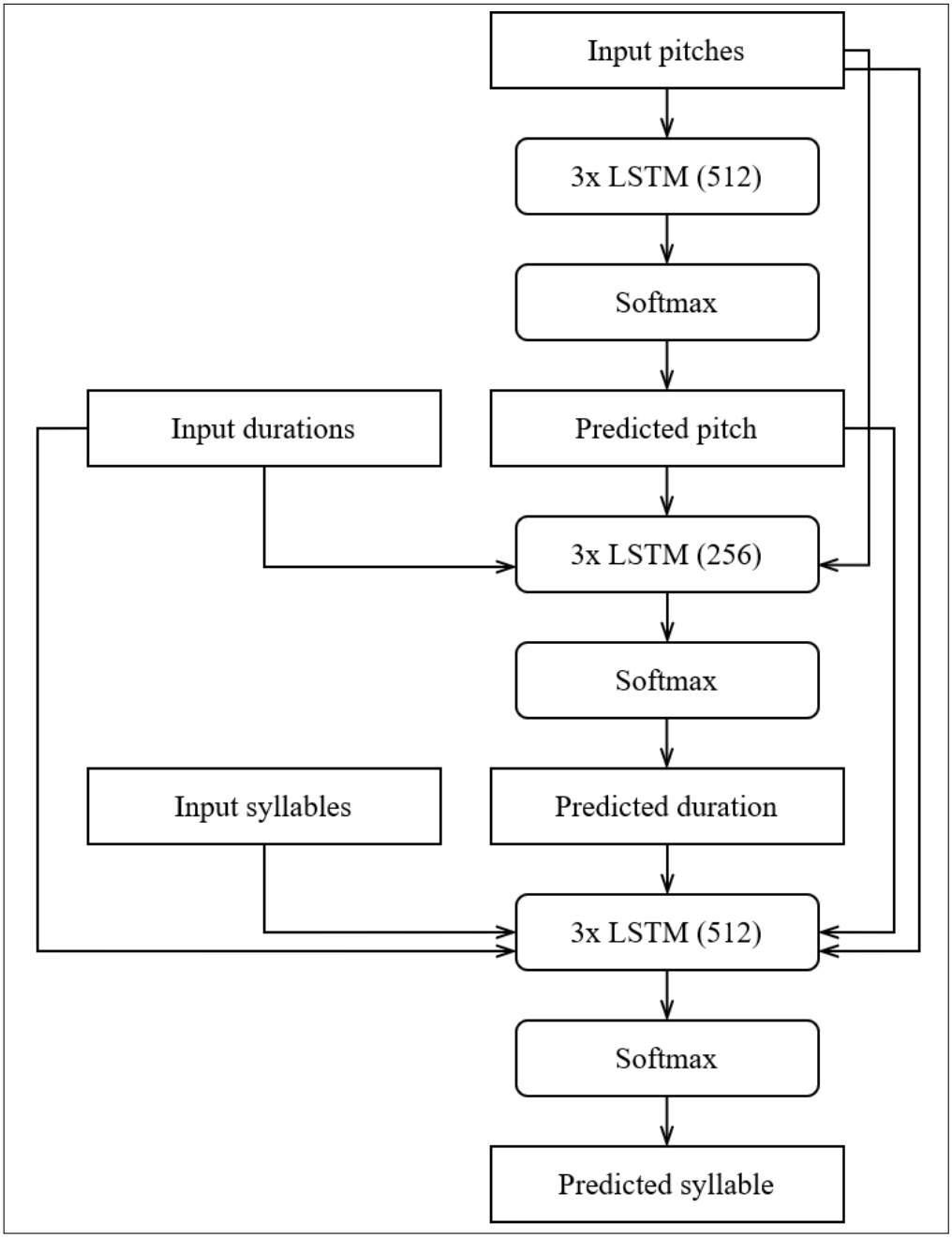

Figure 3.1: Overview of the neural network. The model first generates one pitch, then one duration conditioned on the sampled pitch, and finally one syllable conditioned on the sampled pitch and duration. The squared rectangles represent the input and output layers of the network whereas the rounded rectangles represent the trainable layers. Furthermore, the first number in every LSTM block refers to the number of layers and the second one indicates the amount of units in each layer. Finally, *Softmax* indicates a dense layer with a softmax activation function.

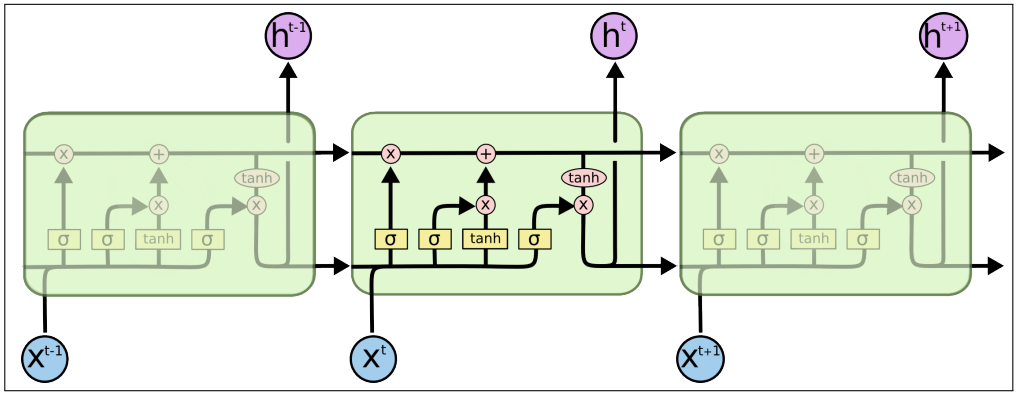

Figure 3.2: Overview of the internal structure of a LSTM layer unrolled in time: a chain of repeating modules of neural networks each with four layers called gates. $\mathbf{x}^{(t)}$ is the input and $\mathbf{h}^{(t)}$ is the hidden state. The figure is taken from [29] and slightly adapted.

as input a sequence of notes of length L, each one-hot encoded, representing the time steps $[t - L, t]$. Three consecutive LSTM layers with 512 units each process this input. Figure 3.2 shows the internal structure of every LSTM layer: a chain of repeating modules of L neural networks each with four layers called gates, interacting between each other. Every module, depicted in Figure 3.3, has a so-called *state*, divided in two vectors: the short-term state $\mathbf{h}^{(\mathbf{t})}$, called *hidden*, and the long-term state $\mathbf{c}^{(\mathbf{t})}$, called *cell*. These represent the memory of the network, i.e the past knowledge that the network currently holds at a given time step t. As first step, the forget layer is used to decide the information to discard from the cell state and it is calculated as follows:

$$\mathbf{f}^{(t)} = \sigma(\mathbf{U}_f^T \mathbf{x}^{(t)} + \mathbf{W}_f \mathbf{h}^{(t-1)}) \tag{3.1}$$

Where σ represents the sigmoid function, $\mathbf{U}$ stands for the weights of the input $\mathbf{x}^{(t)}$ and $\mathbf{W}$ represent the weight of the hidden state of the previous time step $\mathbf{h}^{(t-1)}$. Afterwards, the input gate is used to decide the new information to store in the cell state. This is obtained by first using a sigmoid layer, which decides the value to update, and then by using a tanh layer called input-modulation gate, which creates a vector of new candidate values that could be added to the state. These are calculated as follows:

$$\mathbf{i}^{(t)} = \sigma(\mathbf{U}_i^T \mathbf{x}^{(t)} + \mathbf{W}_i \mathbf{h}^{(t-1)}) \tag{3.2}$$

$$\tilde{\mathbf{c}}^{(t)} = tanh(\mathbf{U}_{\tilde{c}}^T \mathbf{x}^{(t)} + \mathbf{W}_{\tilde{c}} \mathbf{h}^{(t-1)}) \tag{3.3}$$

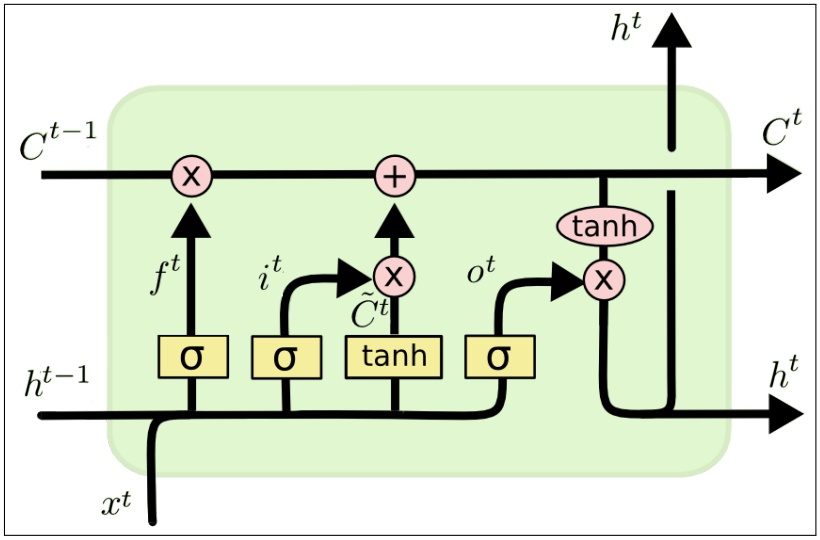

Figure 3.3: Detailed view of the LSTM module. The figure is taken from [29] and slightly adapted.

As third step, the old state cell $\mathbf{c}^{(t-1)}$ is updated to the new state cell $\mathbf{c}^{(t)}$ by calculating the element-wise product of the forget gate and the previous internal cell state, then calculate the element-wise product of the input gate and the input modulation gate and then adding the two vectors:

$$\mathbf{c}^{(t)} = \mathbf{f}^{(t)} \odot \mathbf{c}^{(t-1)} + \mathbf{i}^{(t)} \odot \tilde{\mathbf{c}}^{(t)} \tag{3.4}$$

Finally, the output gate is used to decide which part of the cell state to output:

$$\mathbf{o}^{(t)} = \sigma(\mathbf{U}_i^T \mathbf{x}^{(t)} + \mathbf{W}_o \mathbf{h}^{(t-1)}) \tag{3.5}$$

Thus, the current hidden state is calculated by first taking the hyperbolic tangent of the current internal cell state vector and then performing element-wise product with the output gate:

$$\mathbf{h}^{(t)} = tanh(\mathbf{c}^{(t)}) \odot \mathbf{o}^{(t)} \tag{3.6}$$

The output is then fed to a dense layer with a softmax activation function giving the probability distribution of pitches at the next timestep $t+1$. The network concatenates the input pitches and the predicted pitch with the sequence of corresponding input durations. This is fed to three other LSTM layers with 256 units each. A dense layer with a softmax activation function predicts the duration at time $t+1$. Finally, the network concatenates this output with the input durations, the predicted pitch at time $t+1$, the input pitches, and the corresponding input sequence of syllables and then feds it to three other LSTM

layers with 512 units each. One final dense layer with a softmax activation function outputs the probability distribution of the syllable at the next time step $t + 1$.

Distributing the generation according to this representation ensures that the model first generates one pitch, then one duration conditioned on the sampled pitch, and finally one syllable conditioned on the sampled pitch and duration. This fulfills Condition 1 as indicated in Section 1.1.

A snippet of the code used to create the network is presented below. Only three LSTM layers instead of nine are shown for clarity.

```
note_input = Input(shape=(None,notes_vocab_size)) #
 ↪  the network takes as input a sequence of notes,
 ↪  each one hot encoded. The shape becomes:
 ↪  (batch_size, sequence_length, notes_vocab_size)
note_features = LSTM(512, return_sequences=True,
 ↪  recurrent_dropout=0.3)(note_input) # defines a
 ↪  LSTM layer with 512 units and 30% of dropout
 ↪  chance
note_out = TimeDistributed(Dense(notes_vocab_size,
 ↪  activation='softmax'))(note_features) # define
 ↪  a dense layer with a softmax activation
 ↪  function giving the probability distribution
 ↪  for each of the tokens at the next time step

duration_input =
 ↪  Input(shape=(None,duration_vocab_size))
x_1 = concatenate([note_input, note_out,
 ↪  duration_input]) # concatenate the note input,
 ↪  note output and duration input and use it as
 ↪  input for the next LSTM layer
duration_features = LSTM(256,
 ↪  return_sequences=True,
 ↪  recurrent_dropout=0.3)(x_1)
duration_out =
 ↪  TimeDistributed(Dense(duration_vocab_size,
 ↪  activation='softmax'))(duration_features)

syll_input = Input(shape=(None, syll_vocab_size))
x_2 = concatenate([note_input, note_out,
 ↪  duration_input, duration_out, syll_input])
```

```
syll_features = LSTM(512, return_sequences=True,
 ↪  recurrent_dropout=0.3)(x_2)
syll_out = TimeDistributed(Dense(syll_vocab_size,
 ↪  activation='softmax'))(syll_features)

model = Model(inputs=[note_input, duration_input,
 ↪  syll_input], outputs=[note_out, duration_out,
 ↪  syll_out]) # create the model with the three
 ↪  inputs and three outputs defined above
```

3.2 Training

The dataset this work uses for training is from [7],[1] and is a merging of 7998 MIDI files from the *LMD-full* dataset [30] with 4199 MIDI files from the *Reddit MIDI dataset* [31], with lyrics aligned on a syllable level. This dataset has 3,225,213 pairs of syllables with MIDI note-on events. For example, a short excerpt from the song "Listen to the rhythm of the falling rain" is represented as follow:

```
List    [74.0, 0.5, 0.0]
en      [72.0, 1.0, 0.0]
to      [72.0, 0.5, 0.0]
the     [69.0, 1.0, 0.0]
rhy     [69.0, 0.5, 0.0]
thm     [67.0, 1.0, 0.0]
```

The MIDI attributes on the right-hand side are, in order: the pitch of the note, the duration of the note and the duration of the rest before the note.

In order to reduce the computational complexity and to unify the data under the same domain, I applied several preprocessing steps to the syllables including stripping, converting to lowercase, and removing punctuation symbols and numbers. Furthermore, I replaced the tokens appearing less than 10 times with the unique symbol <*xxx*>, reducing the syllable space from 13904 to 5224. Likewise, I considered only pitches between C1 and B7, and therefore the pitch space is reduced from 155 to 84. Finally, I kept only durations that are multiples of a demiquaver, up to a semibreve. This reduces the size of the duration vocabulary from 19 to 10.

[1] Name: *lmd-full_and_reddit_MIDI_dataset*

I then divided the input MIDI-syllable pairs into sequences of length L, in the same way as [7] do. When the number of notes in a song file [2] file is not divisible by L, I do not consider the remainder of the division. I also ignore a song file if it has less than L notes. As result, the number of training sequences of each length are: 155,158 ($L = 20$), 47,815 ($L = 60$), and 526 ($L = 500$). I train the network for 100 epochs using a batch size of 128 and categorical crossentropy loss. Furthermore, inspired by the work of Colombo and Gerstner [3], I perform dropout between layers at a rate of 0.3 to prevent overfitting and improve the performance of the network [32].

3.3 Generation

Once the network is trained, the generation process is iterative: I feed the network one pitch with the corresponding duration and syllable and it outputs the probability distribution for the next set of attributes. I then sample from the probability distribution according to a temperature hyper-parameter T. Low temperature such as $T = 0$ is equivalent to selecting the token at the mode of the distribution. Opposed to this, higher temperature such as $T = 1$ increases the sensitivity towards low probabilities candidates, resulting in more diversity and "creativity". I set $T = 1$ when sampling the distribution of notes and durations and to 0 when sampling the distribution of syllables, resulting in a more creative melody with a more coherent and meaningful text. I also test the case of $T = 1$ when sampling the syllables distribution. Examples can be seen in Chapter 4.

Once sampled, I give the attributes as input to the network for the next time step and the process is iterated until the desired sequence length is reached. With the exception of computational resources, there are potentially no limits on the output sequence length and therefore Condition 2 is fulfilled.

Finally, it is important to point out that it is possible to condition the generation process to a manual input by feeding the network as first step one or more set of pitches, durations and syllable chosen by the user. This fulfills Condition 3 as indicated in Section 1.1.

3.4 Evaluation metrics

Most of the evaluation metrics used in this work are adopted from [7] and are already mentioned in Section 2.2. The pitch, duration and interval distribution

[2] Each of the song in the dataset is stored as a *.npy* file containing the MIDI-syllable pairs.

highlight how well the network is able to successfully capture the transition probabilities of the considered musical attributes. Furthermore, the N-gram distribution outline if, and to what extent, repetitions of musical figures such as motifs and arpeggios are present in the generated sequences. Finally, the number of unique MIDI numbers, MIDI number span, and song length give a further indication of the ability of the network to create sensible material.

As an extension of those metrics, I also evaluate how often the generation of consecutive syllables form sensible words. Let S be the generated syllable sequence, and denote by $|\hat{S}|$ the number of syllables in S that do not form real words with its sequence neighbors. This is computed by iteratively checking if, starting from every syllable in the sequence, it is possible to form a word present in an English dictionary by concatenating consecutive syllables. In order to account the cases where the generated sequences consists in the repetition of few syllables that can have word meaning (eg: *you*, *can*), I introduce a normalization measure. Denote by $|SU|$ the unique set of S. The proportion of syllables that do not form a word is:

$$S_{norm} = \frac{|\hat{S}|}{|SU|} \tag{3.7}$$

Using this metric is possible to evaluate how precise is the model in the generation of sensible words by combining different consecutive syllables, and therefore it is possible to observe whether the model is able to learn sensible linguistic properties or not.

Chapter 4

Results

I build networks modeling sequences of the three different lengths. At first I selected $L = 20$ and then I explored intermediate values between 20 and 60. The preliminary experimental results showed similar outcomes for these lengths. I also decided to test the extreme case of $L = 500$. Once I trained the three networks, I used them to generate 500 samples each and then evaluated them using the metrics defined in section 3.4. In a similar fashion, I defined the baseline by computing the same metrics on the sequences of the training set.

Figure 4.1 shows the training loss for $L = 20$ and $L = 60$, while $L = 500$ is excluded for readability. As expected, with low values of L the training is smooth and the cross-entropy loss is constantly reduced. Furthermore, regardless of the value of L, we can observe that the syllable loss starts higher than the note and duration one and it generally requires a higher amount of iterations to converge. This is due to the fact that the syllable space is much larger than the notes and duration ones. In the case of sequences length equal to $L = 500$, the training is far slower in terms of convergence, as expected.

Figure 4.2 and Figure 4.3 shows several examples of generated samples, converted to staff notation. We can observe that they are all playing in common scales and therefore do not sound dissonant, but a deeper musical evaluation is needed. This is beyond the scope of this master thesis and is left as future work. More samples are available at `http://seicaratteri.github.com/paper/samples.html`.

The following sections evaluate the quality of the generated samples by first using quantitative metrics and then by discussing the use of the model in a real case scenario.

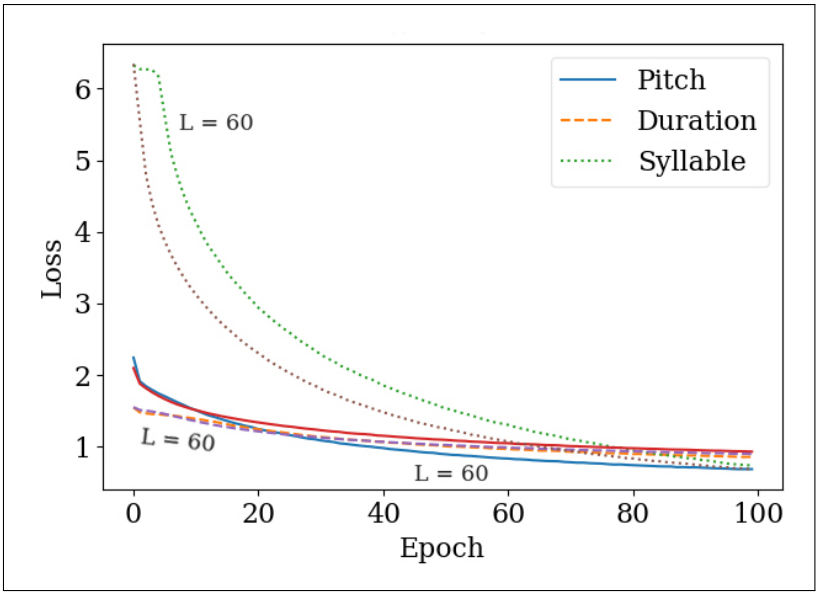

Figure 4.1: Training losses for $L = 20$ and $L = 60$. $L = 500$ is excluded for readability.

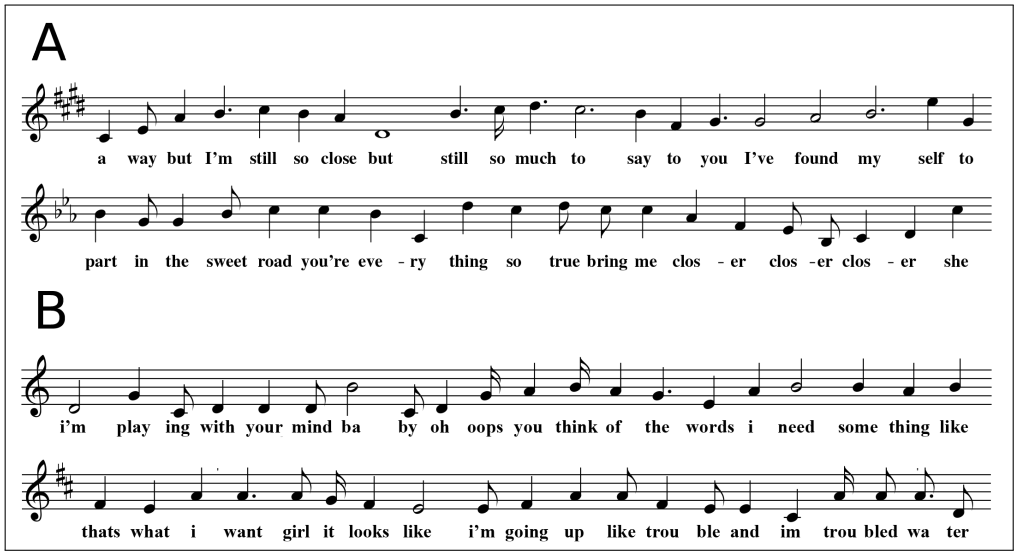

Figure 4.2: Four example outputs of the system generating sequences of length $L = 20$. The samples of **A** are generated by sampling from the syllables distribution with temperature $t = 0$, whereas the samples of **B** use $t = 1$.

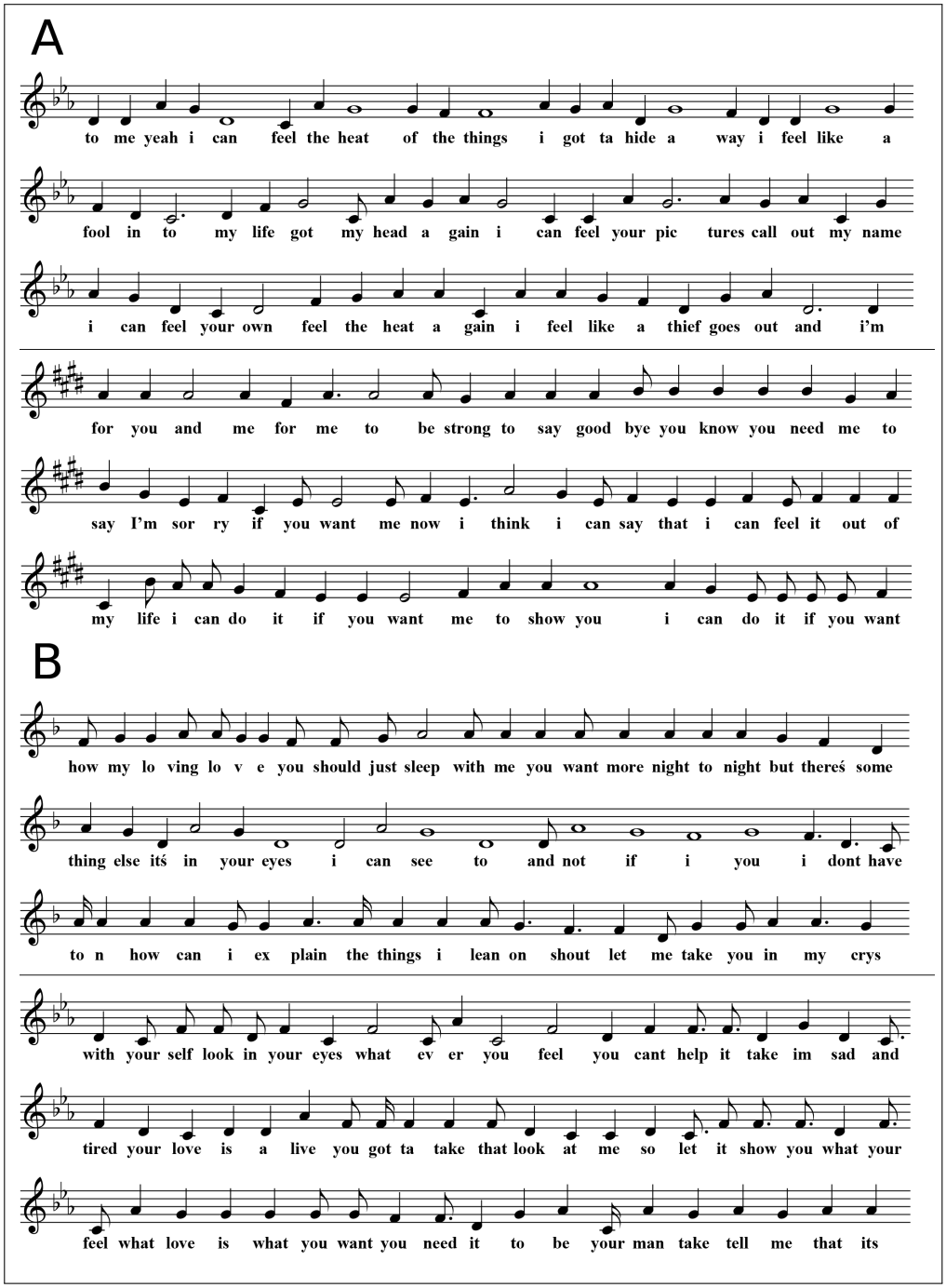

Figure 4.3: Four example outputs of the system generating sequences of length $L = 60$. The samples of **A** are generated by sampling from the syllables distribution with temperature $t = 0$, whereas the samples of **B** use $t = 1$.

4.1 Numerical statistics comparison

The pitch distribution of the generated samples for the three sequence lengths resembles the distributions of the baseline: most of the sampled pitches are in the range $[55, 81]$. This can be further observed by comparing the interval [1] distribution of the generated samples and the baseline, depicted in Figure 4.4 and Figure 4.5: for all the three considered sequence lengths, the distributions closely resembles the baseline, and perfect unison, major second and minor third appear as the most common intervals. Moreover, the network can ultimately predict a note with an accuracy of 69.85% for $L = 20$, 76.93% for $L = 60$ and 87.76% for $L = 500$.

Figure 4.6 and Figure 4.7 show the duration distributions for $L = 20$, $L = 60$ and $L = 500$. As it can be seen, the distributions of the generated samples results very similar to the baseline ones, indicating that the model successfully captured the transition probabilities between durations. Indeed, the network can predict a duration with an accuracy of 68.70% for $L = 20$, 69.52% for $L = 60$ and 70.62% for $L = 500$.

The N-grams distributions, depicted in Figure 4.8 and Figure 4.9, present generally lower values compared to the baseline, and this is especially true in the case of high N-grams values such as 4 and 5. This results in the presence of fewer repetitions of motifs and arpeggios in the generated sequences and therefore less melodic coherence over long time spans.

Table 4.1 shows a summary of the numerical statistics. As it can be seen, the models trained with $L = 20$ and $L = 60$ very closely resemble the baseline, showing once more that the network is able to produce similar material to the training data. Furthermore, the S_{norm} value indicates that the network is able to successfully generate consecutive syllables that form meaningful words, indicating that the model learned coherent linguistic properties. In the case of $L = 500$ the trained model presents a higher variance in all the numerical metrics compared to the baseline and especially in the case of S_norm we can observe that the value is more than 30 times higher. This reflects on the accuracy of the syllables prediction: it is as low as 32.18%, indicating that the network in the case of fewer training data and longer sequences is not able to successfully capture the transition probabilities of syllables.

[1] In music theory, an interval is the difference in pitch between two consecutive notes [33].

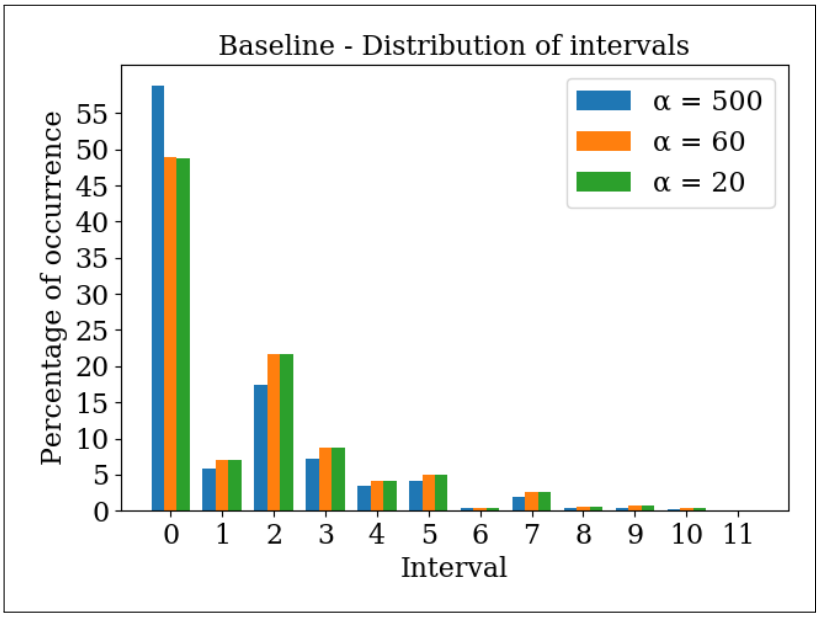

Figure 4.4: Baseline interval distribution for the three L considered.

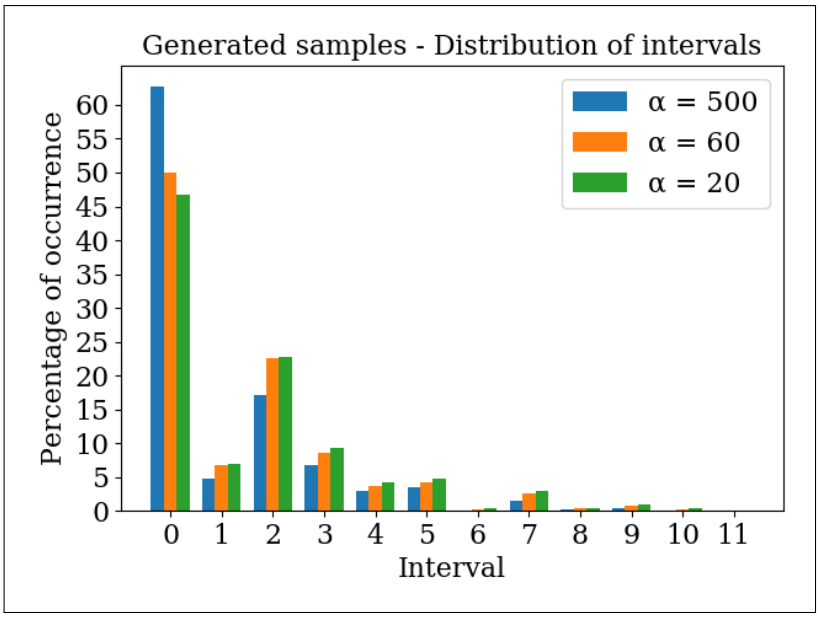

Figure 4.5: Generated samples interval distribution for the three L considered.

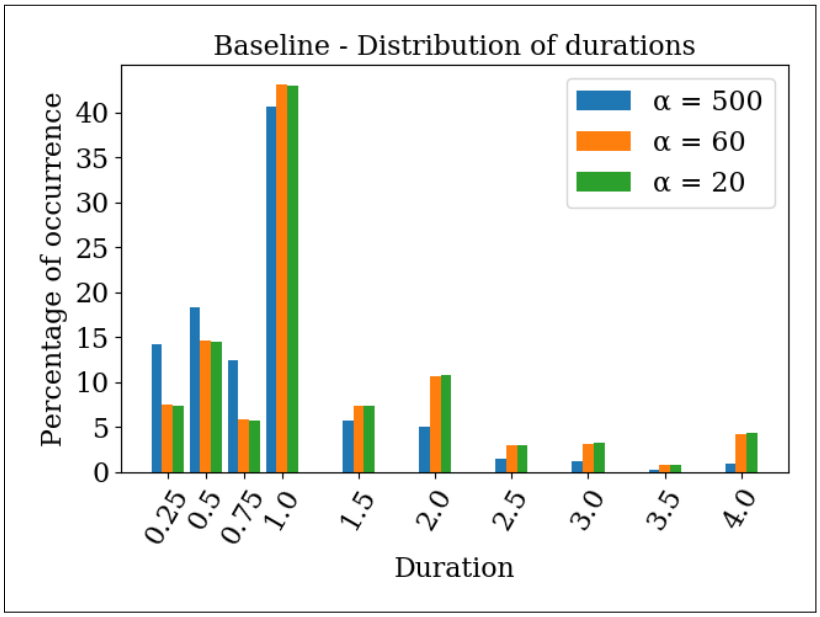

Figure 4.6: Baseline duration distribution for the three L considered.

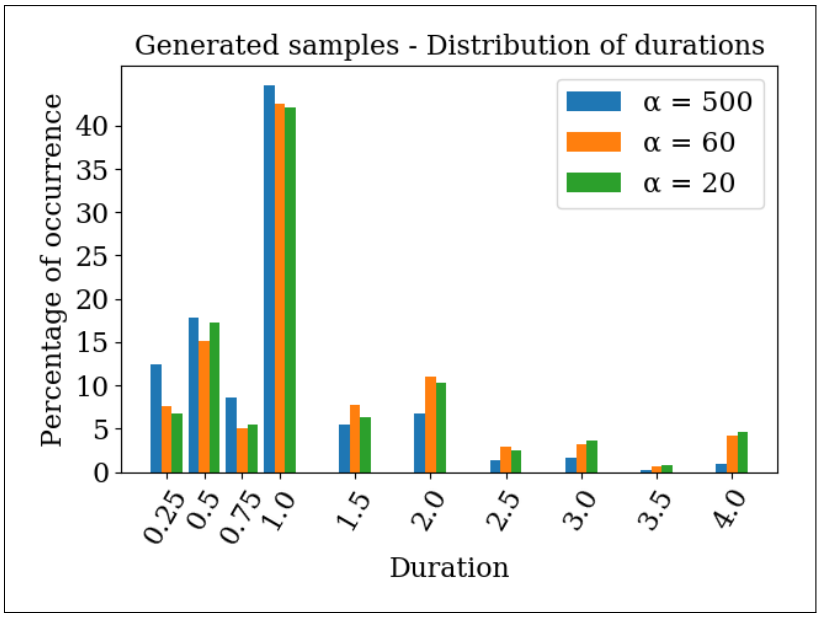

Figure 4.7: Generated samples duration distribution for the three L considered.

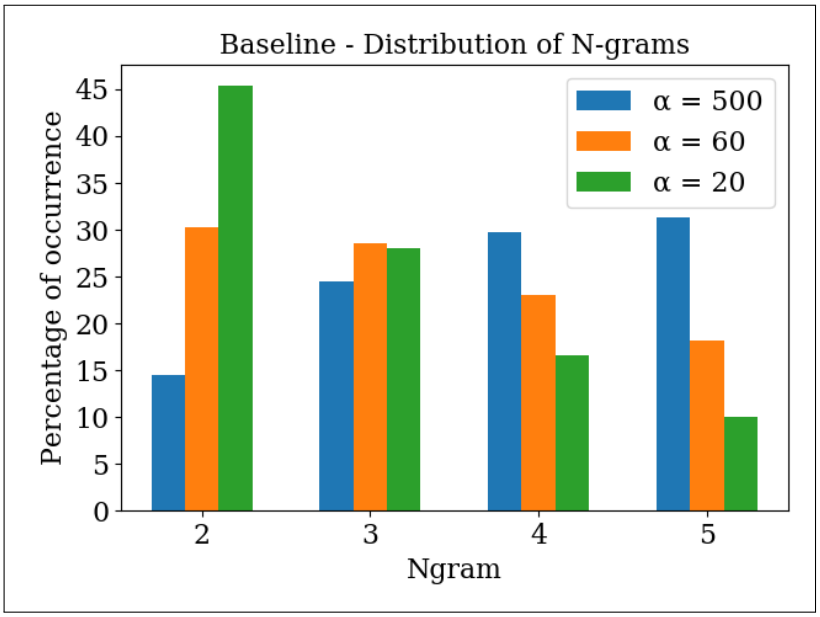

Figure 4.8: Baseline N-gram distribution for the three L considered.

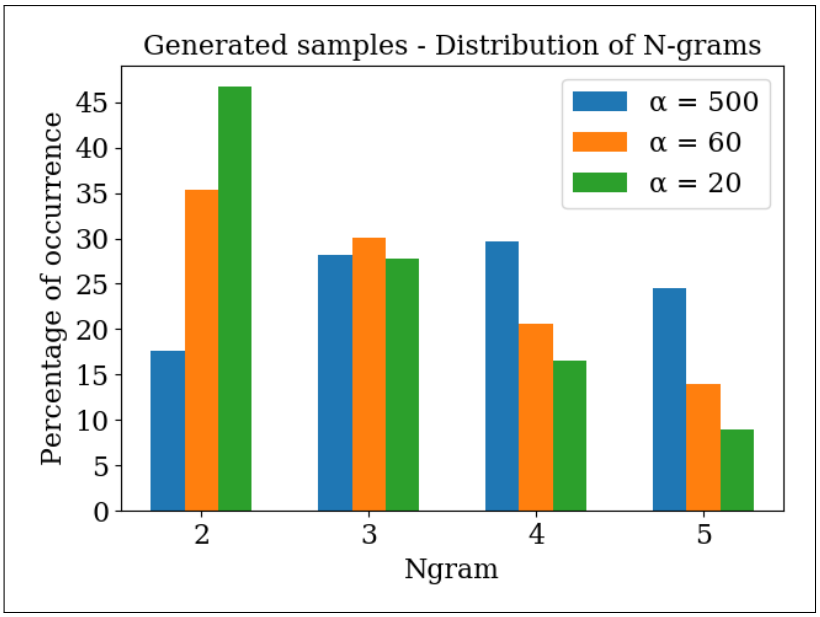

Figure 4.9: Generated samples N-gram distribution for the three L considered.

	L	MIDI number span	Unique MIDI numbers	Song length	S_{norm}
baseline	20	26.658	5.869	76.682	0.444
model	20	26.836	5.822	75.862	0.645
baseline	60	31.406	8.344	229.831	0.540
model	60	32.028	8.410	231.472	0.816
baseline	500	31.074	12.470	1931.7	2.244
model	500	36.38	15.91	2011.57	67.54

Table 4.1: Summary of the numerical statistics computed on the training set (*baseline*) and the generated sequences (*model*), for three different sequences length L. *MIDI number span* is the average of the difference between the highest and the lowest MIDI number in the sequences. *Unique MIDI numbers* is the average of the count of how many unique pitches are present in the sequences. *Song length* is the average of the sum of all the durations in the sequences. S_{norm} is the average of the proportions of syllables that do not form a word in the sequences.

4.2 Use of the models in the AI Song Contest

As further evaluation of the method, I used the network as part of a human-AI collaboration which aimed to create a song entry for the VPRO AI Song Contest [34] [2]. I had the network generate a bulk of 100 sequences and then I found the most interesting material by listening to them. Some sequences were used as fundamental building blocks for the song, as shown in Figure 4.10: the bassline is the sequence number 0 from the samples generated with sequence length $L = 20$, whereas the lead melody is the sequence number 59 from the samples generated with $L = 60$. Furthermore, the spoken text in the intro and in the bridge comes from the sequence number 8 from the samples generated with $L = 30$. Starting from those, I composed all the remaining elements such as the chord progression, the drums, and the accompanying instrument, and as the final step I arranged and mixed them. The entry for the contest can be listened at `https://bit.ly/3ceBfcu`, and all the generated samples can be found at `https://bit.ly/3fbCDyA`.

[2] The AI song contest is an unofficial spin-off of the Eurovision contest organised by the Dutch broadcast organization VPRO, in which the participants are asked to generate an Eurovision-styled song using artificial intelligence for the greatest extent possible. More information can be found at: `https://www.vprobroadcast.com/titles/ai-songcontest.html`.

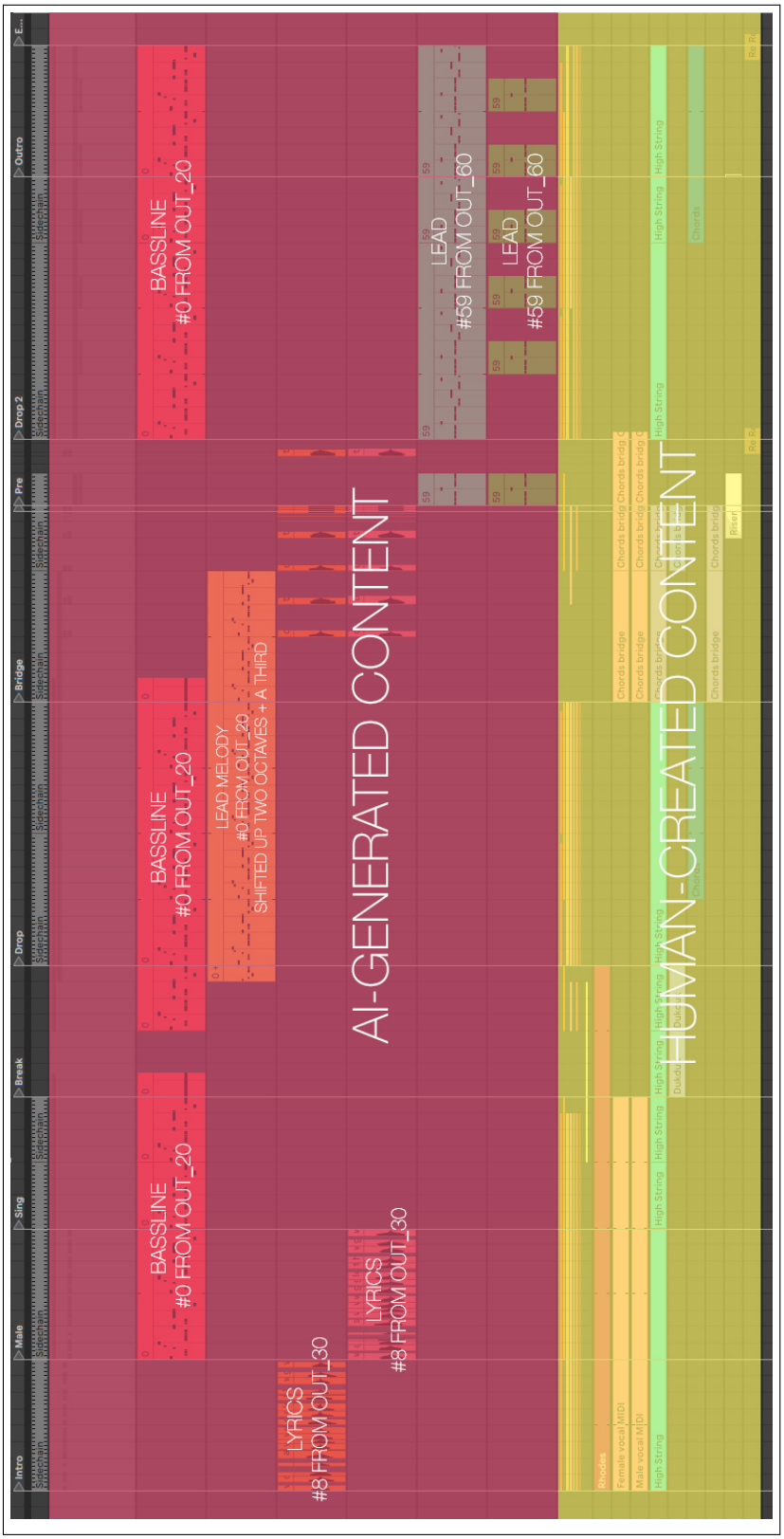

Figure 4.10: Detail of the division between AI-generated content and human-created content in the song used as entry for the VPRO AI Song Contest.

Chapter 5

Discussion

The results outlined in the previous section show that the network for the sequence length $L = 20$ and $L = 60$ is able to approximate well the distribution of pitches, durations and syllables and indeed they all closely resemble the training data. However, the N-gram distribution highlights the lack of recurring musical figures such as motifs and arpeggios and this results in generally less pleasing melodies compared to the human-composed ones found in the training set. The example below show a recurring note pattern from a randomly-picked melody in the dataset:

G5 G5 G5 G#5 **A#5 A#5 C6 G5 G5 G5 F5 G5 A#5 A#5 C6 G5 G5 G5 F5 G5** G5 G5 G#5 **A#5 A#5 C6 G5 G5 G5 F5 G5 A#5 A#5 C6 G5 G5 G5 F5 G5** G5 C5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 G5 F5 G5 G5 G5 A#5 D6 G5 G5 G5 G#5 **A#5 A#5 C6 G5 G5 G5 F5 G5 A#5 A#5 C6 G5 G5 G5 F5 G5** G5 G5 G#5 **A#5 A#5 C6 G5 G5 G5 F5 G5 A#5 A#5 C6 G5 G5 G5 F5 G5** *[...]*

I observed this as well when I used the network to generate material for the VPRO AI song contest [34]. Even if some of the sequences were good enough to be used as foundamental building blocks for the song, most of them were neither sufficiently musically or grammatically coherent to be used in a real case scenario without some manual human adjustment. This is caused by a variety of factors: first of all, the syllabic generation for its very nature can more easily lead to imperfect texts because even a single wrong syllable would lead to break word-level meaning. This assumes a even deeper implication in the case of concurrent generation of melody and lyrics. Take, for instance, the case in which the generated melody has a formally correct and pleasant musical structure. If even just one of the syllable in the text is wrong, then the

whole lyric would lose meaning and therefore the whole generated sequence would not be suitable for a real case application. Naturally, the opposite scenario is also possible: the lyrics could be meaningful but if the melody is not, then the whole sequence would not usable.

Furthermore, another problem arise from the dataset used. The network learns to approximate the distribution of the data in the dataset and naturally, if this contains mistakes, then the network learns to reproduce them as well. I performed an exploratory analysis and noticed that several songs in the dataset contain incorrect lyrics, such as the one shown below:

```
- File name: 692a2521aa9ed00d78453271ef707041.npy
- Song name: Engelbert Humperdinck - The last waltz

- Dataset lyrics:

I wond ered I or should stay
The had ly one more to And then I saw
out cor ner
my eyes
A lit girl lone so I had Last Waltz with
[...]

- Correct lyrics:

I wondered should I go or should I stay
The band had only one more song to play
And then I saw you out the corner of my eyes
A little girl alone and so shy
I had the last waltz with you
[...]
```

We can observe that several words are missing and therefore the lyrics lose meaning and grammatical coherence. This is due to the fact that the authors of the dataset performed the automatic parsing of syllable-note pairs without manual annotation and verification [1].

Furthermore, we can notice a similar behaviour in some of the generated sequences as well, such as:

[1] I communicated the errors I found to the authors of the dataset, and they are now aware of the problem. Two more examples of incorrect text are attached in Appendix A.

```
I'll be waiting end er of salt your
my mother could er i don't ever see me
```

It is hard to say whether the absence of meaning and grammatical structure in the considered sample is a failure of the model or a consequence of the noisy dataset. Nonetheless, it is possible to argue that the network is able to approximate well the distribution of pitches, durations and syllables, and therefore is plausible that the model learned to create incorrect lyrics because of the nature of the training data.

It would be interesting to investigate the performance of the model trained on an updated dataset, where the mistaken samples are either removed or corrected. In order to do so, it would be necessary to first identify the faulty songs, for example by calculating the S_{norm} value for each of the samples in the training set, and by marking the ones exceeding a pre-defined threshold. The marked songs could then be either removed or corrected by aligning the syllables to the the timestamp information of the MIDI tracks, or by employing the Needleman-Wunsch algorithm [35], in a similar fashion to what done by [6]. This was not done in this dissertation due to lack of time, and is left as future work.

Regardless of the limitations discussed above, the proposed method could be used in a human-AI Collaboration and therefore proves its potentiality as a part of a creative pipeline. Indeed, the sequences generated by the system sparked my creativity and brought to the creation of a song that I probably would not have envisioned otherwise. Therefore, I can imagine the model as an assistant for musicians, augmenting the process of music creation. Furthermore, with a larger variety of data available, the model could be trained for different styles and languages and therefore it could fill specific roles in music production.

Chapter 6

Conclusions

In this dissertation, I presented a neural network capable of generating lyrics and melody concurrently. The proposed architecture extends the work of [2, 3] and uses a similar RNN model with the addition of conditioning on the syllables. The evaluation metrics validated the output sequences of the network, showing that the generated material in terms of pitches, durations, and intervals closely resembles the training data. Furthermore, the method was applied in a real case scenario and it could generate material used as building blocks of a song for the VPRO AI Song Contest [34]. At the same time, this highlighted the limitations of the system: it can be a useful tool to augment the creative process of musicians, but it can not replace them.

Several future works can further improve the performance of the network. One way could be to narrow down the problem: the dataset used spans over a wide variety of different genres and styles. With more data available, the network could be trained for one specific genre, and this could result in the generation of more coherent sequences.

Another possible direction could be investigating the use of a CNN-GAN, similar to what proposed in [4]. The data would be represented as a piano roll with a three dimensional matrix where the x axis represent time, the y axis represent the pitch, and the z axis represent the syllable. As the syllable space is very large, it could be convenient to use a lower-dimensional representation of it, such as the embedding learned with a skip-gram model [23]. CNN-GANs architectures proved to be successful in the past [4, 36, 37] and they might have the potential to be applied as well to the problem of concurrent generation of melody and lyrics.

Furthermore, it would be interesting to compare the proposed method to the ones that generate melody conditioned to the lyrics and vice versa, in order

to investigate whether the concurrent generation can more accurately capture the relations between notes and text or not.

Moreover, as mentioned in Chapter 5, it would be interesting to train the model on a corrected version of the dataset in order to evaluate how much the faulty nature of the training data is influencing the results.

Finally, I have been doing some preliminary work in gathering and applying the method to a Gregorian chant datasets. I believe that the intrinsic relation between music and text of this genre would make it an ideal application for the proposed method. I scraped the melodies and lyrics from [38] and tried to align notes and syllables. This was possible in the majority of the cases, but not in all: indeed, the 29.4% of the words in the dataset, split in syllables, are associated with an unequal number of notes and therefore a direct alignment is not possible. Some special logic is needed to handle these cases, and this is left as future work.

Bibliography

[1] B. L. Sturm et al. “Music Transcription Modelling and Composition Using Deep Learning”. In: *Proc. Conf. Computer Simulation of Musical Creativity*. Huddersfield, UK, 2016.

[2] F. Colombo, A. Seeholzer, and W. Gerstner. “Deep Artificial Composer: A Creative Neural Network Model for Automated Melody Generation”. In: *Proc. EvoMUSART*. 2017.

[3] Florian Colombo and Wulfram Gerstner. “Bachprop: Learning to compose music in multiple styles”. In: *arXiv preprint arXiv:1802.05162*. 2018.

[4] Hao-Wen Dong et al. “Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[5] Eric Malmi et al. “Dopelearning: A computational approach to rap lyrics generation”. In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2016.

[6] Kento Watanabe et al. “A melody-conditioned lyrics language model”. In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. 2018.

[7] Yi Yu and Simon Canales. “Conditional LSTM-GAN for Melody Generation from Lyrics”. In: *arXiv preprint arXiv:1908.05551*. 2019.

[8] Joseph Rothstein. *MIDI: A Comprehensive Introduction*. USA: A-R Editions, Inc., 1992. ISBN: 0895792583.

[9] Bob LT Sturm et al. “Artificial intelligence and music: open questions of copyright law and engineering praxis”. In: *Arts*. Multidisciplinary Digital Publishing Institute. 2019.

[10] Andre Holzapfel, Bob Sturm, and Mark Coeckelbergh. “Ethical dimensions of music information retrieval technology”. In: *Transactions of the International Society for Music Information Retrieval*. Ubiquity Press, 2018.

[11] Torsten Anders and Eduardo R Miranda. “Constraint programming systems for modeling music theories and composition”. In: *ACM Computing Surveys (CSUR)*. ACM New York, NY, USA, 2011.

[12] Arne Eigenfeldt and Philippe Pasquier. “Realtime generation of harmonic progressions using controlled Markov selection”. In: *Proceedings of ICCC-X-Computational Creativity Conference*. 2010.

[13] Stephen Davismoon and John Eccles. “Combining musical constraints with Markov transition probabilities to improve the generation of creative musical structures”. In: *European Conference on the Applications of Evolutionary Computation*. Springer. 2010.

[14] Jose D Fernández and Francisco Vico. “AI methods in algorithmic composition: A comprehensive survey”. In: *Journal of Artificial Intelligence Research*. 2013.

[15] Cheng-Zhi Anna Huang et al. “Music Transformer: Generating Music with Long-Term Structure”. In: *arXiv preprint arXiv:1809.04281*. 2018.

[16] Prafulla Dhariwal et al. “Jukebox: A Generative Model for Music”. In: *arXiv preprint arXiv:2005.00341*. 2020.

[17] Sara Feldman. “Co-creation: human and AI collaboration in creative expression”. In: *Electronic Visualisation and the Arts (EVA 2017)*. 2017.

[18] Matthew Guzdial and Mark Riedl. “An interaction framework for studying co-creative ai”. In: *arXiv preprint arXiv:1903.09709*. 2019.

[19] Marcelo LaFleur. “Frontier Issues: The impact of the technological revolution on labour markets and income distribution”. In: *Department of Economic & Social Affairs, UN, accessed*. 2017.

[20] Benjamin Wardhaugh. *Music, Experiment and Mathematics in England, 1653-1705*. Ashgate. ISBN: 978-1-351-55708-5.

[21] Adam Alpern. “Techniques for algorithmic composition of music”. In: *On the web: http://hamp. hampshire. edu/adaF92/algocomp/algocomp*. 1995.

[22] Francesca Rossi, Peter van Beek, and Toby Walsh. *Handbook of Constraint Programming (Foundations of Artificial Intelligence)*. USA: Elsevier Science Inc., 2006. ISBN: 0444527265.

[23] Tomas Mikolov et al. “Efficient Estimation of Word Representations in Vector Space”. In: *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*. 2013.

[24] *ABC notation*. http://abcnotation.com/. Accessed: 2020-06-03.

[25] Alec Radford et al. “Language models are unsupervised multitask learners”. In: *OpenAI Blog*. 2019.

[26] David Donahue and Anna Rumshisky. “Adversarial text generation without reinforcement learning”. In: *arXiv preprint arXiv:1810.06640*. 2018.

[27] François Chollet et al. *Keras*. https://keras.io. Accessed: 2020-06-08. 2015.

[28] Martın Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[29] *Understanding LSTM Networks*. https://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2020-06-03.

[30] *Lakh MIDI Dataset*. https://colinraffel.com/projects/lmd/. Accessed: 2020-04-15.

[31] *The largest MIDI collection on the internet*. https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/. Accessed: 2020-04-15.

[32] Yarin Gal and Zoubin Ghahramani. “A theoretically grounded application of dropout in recurrent neural networks”. In: *Advances in neural information processing systems*. 2016.

[33] Ebenezer Prout. *Harmony : its theory and practice / by Ebenezer Prout*. English. Rev. and largely rewritten. AMS Press New York, 1971. ISBN: 0404051448.

[34] *VPRO AI Song Contest description*. https://www.vprobroadcast.com/titles/ai-songcontest.html. Accessed: 2020-04-14.

[35] S. B. Needleman and C. D. Wunsch. “A general method applicable to the search for similarities in the amino acid sequence of two proteins”. In: *J. Mol. Biol.* 1970.

[36] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein gan”. In: *arXiv preprint arXiv:1701.07875*. 2017.

[37] Alec Radford, Luke Metz, and Soumith Chintala. “Unsupervised representation learning with deep convolutional generative adversarial networks”. In: *arXiv preprint arXiv:1511.06434*. 2015.

[38] *Cantus index melodies*. `http://cantusindex.org/melodies`. Accessed: 2020-06-05.

Appendix A

Incorrect lyrics in the dataset

The following represent two further examples of incorrect lyrics found in the dataset, as discussed in Section 5.

```
*** 2 ***

- File name: d483c4a0cd8047bda9689efbd89c6283.npy
- Song name: Mr. President - Simbaleo

- Dataset lyrics:

Tell who's in the chen tell me
tell me who is scratch ing my Yeah!
It's a cub in the kit tell me
tell me would you put me the Yeah!
I walk through the jungle
[...]

- Correct lyrics:

Tell me who's in the kitchen? Who is scratching at my door?
It's the cat in the kitchen, kittie, pretty on the floor

Ha, ha, ha, ha
I was walking thru the jungle in the summertime
[...]
```

*** 3 ***

- File name: 7d4791e986432c2ded339aa3d5215afe.npy
- Song name: Phil Collins - Do You Remember

- Dataset lyrics:

We ne ver talked bout But I hear the was
I'd call you up to say sor But I n't
want to your 'Cos I love but I can't talk
a more
There's a I des in your Yes could like tried
be fore
when you kept on ling those Do re ber?
[...]

- Correct lyrics:

We never talked about it
But I hear the blame was mine
I'd call you up to say I'm sorry
But I wouldn't want to waste your time
'Cause I love you, but I can't take any more
There's a look I can't describe in your eyes
If we could try, like we tried before
Would you keep on telling me those lies?
(Telling me lies)
Do you remember?

Appendix B

ISMIR paper submission

CONCURRENT GENERATION OF MELODY AND LYRICS BY RECURRENT NEURAL NETWORKS

First Author
Affiliation1
author1@ismir.edu

Second Author
Retain these fake authors in submission to preserve the formatting

Third Author
Affiliation3
author3@ismir.edu

ABSTRACT

This paper proposes a conditioned recurrent neural network architecture for concurrent melody and lyrics generation. This is in contrast to methods that first generate music and then lyrics, or vice versa. The system is trained to first sample a pitch from a distribution, then sample a duration conditioned on the sampled pitch, and finally sample a syllable conditioned on the sampled pitch and duration. We show the trained system generates music and text sequences that exhibit some sensible musical and linguistic properties, but more data is needed to improve the results.

1. INTRODUCTION

The generation of a melody with lyrics is a challenging problem in the field of artificial intelligence. In the case of melody generation, musical features such as pitch and duration have to be considered so that the composition follows common musical rules about rhythm and structure. Likewise, lyrics have to respect grammatical structures and in the case of syllable generation, consecutive syllables must form sensible words. Previous works either considered the generation of melody and lyrics as two separate processes [3–5, 7, 8], or the generation of the former conditioned on the latter, and vice versa [9, 10].

This paper proposes modeling melody and lyrics concurrently using a conditioned recurrent neural network architecture similar to [3, 4]. The network first generates a pitch, then conditioned on that pitch generates the corresponding note duration, and finally conditioned on the pitch and duration, generates the syllable. This process is repeated a predefined number of times. To the best of our knowledge, this is the first attempt on such a concurrent generation of notes and lyrics.

Section 2 presents an overview and a brief discussion of related work. Section 3 discusses our proposed system: its architecture, training, and evaluation. Section 4 evaluates the generated samples. Section 5 concludes the paper and proposes possible future improvements. [1]

[1] The source code and the trained models will be publicly available on Github after the paper is reviewed.

2. RELATED WORKS

The last few years have seen a rise in the development of artificial neural networks motivated by their performances in a wide variety of fields. Naturally, they have also been applied to music generation tasks. This includes the generation of folk-music melodies using LSTM units that learn from music transcriptions expressed with a high-level vocabulary [8]. Another work [4] proposes a conditional recurrent neural network that jointly models pitches and durations, effectively taking into account the relation between a note duration and its pitch. A further variation [3] proposes a more general and normalized representation of music. Moreover, trying to extend the generation from monophonic to polyphonic music, another paper [5] trains a CNN-GAN on a piano roll representation and is able to generate five tracks including bass, drums, guitar, piano and strings.

Lyrics are often sung to a melody and thus some research has tried to model and generate lyrics. One [7] worked specifically for rap text generation. From a set of existing lyrics, the neural network learns to combine different lines together to form a coherent text. Other works explore the relationship between melody and lyrics and they either try to generate the former conditioned to the latter, or vice versa. In the case of lyrics conditioned on melody, [9] proposes a recurrent neural network language model able to generate words with a one to one correspondence to MIDI notes. In a similar fashion but for the opposite purpose, [10] uses a LSTM-GAN architecture to generate MIDI conditioned on lyrics on a syllable level.

3. METHOD

Our work focuses on generating melody and lyrics in a concurrent fashion. The architecture, its training and evaluation are now explained in detail.

3.1 Neural network

The model we use is diagrammed in Fig. 1. It is a conditional recurrent neural network similar to that proposed in [3, 4]. The network at time t takes as input a sequence of notes of length L, each one-hot encoded. Three consecutive LSTM layers with 512 units each process this input, and are then followed by a dense layer with a softmax activation function giving the probability distribution of pitches at the next timestep $t + 1$. The network concatenates the input pitches and the predicted pitch with the

sequence of corresponding input durations. This is fed to three other LSTM layers with 256 units each. A dense layer with a softmax activation function predicts the duration at time $t+1$. Finally, the network concatenates this output with the input durations, the predicted pitch at time $t+1$, the input pitches, and the corresponding input sequence of syllables and then feds it to three other LSTM layers with 512 units each. One final dense layer with a softmax activation function outputs the probability distribution of the syllable at the next time step $t+1$. Distributing the generation according to this representation ensures that the model first generates one pitch, then one duration conditioned on the sampled pitch, and finally one syllable conditioned on the sampled pitch and duration.

3.2 Training

The dataset we use for training is from [10],[2] and is a merging of 7998 MIDI files from the *LMD-full* dataset [1] with 4199 MIDI files from the *Reddit MIDI dataset* [2], with lyrics aligned on a syllable level. This dataset has 3,225,213 pairs of syllables with MIDI note-on events. In order to reduce the computational complexity and to unify the data under the same domain, we applied several preprocessing steps to the syllables including stripping, converting to lowercase, and removing punctuation symbols and numbers. Furthermore, we replaced the tokens appearing less than 10 times with the symbol <*xxx*>, reducing the syllable space from 13904 to 5224. Likewise, we considered only pitches between C1 and B7, and therefore the pitch space is reduced from 155 to 84. Finally, we kept only durations that are multiples of a demiquaver, up to a semibreve. This reduces the size of the duration vocabulary from 19 to 10.

We divided the input MIDI-syllable pairs into sequences of length L. When the number of notes in a MIDI file is not divisible by L, we do not consider the remainder of the division. We also ignore a MIDI file if it has less than L notes. The number of training sequences of each length are: 155,158 ($L=20$), 47,815 ($L=60$), and 526 ($L=500$). We train the network for 100 epochs using a batch size of 128 and categorical crossentropy loss. Furthermore, we perform dropout between layers at a rate of 0.3 to prevent overfitting [6].

Figure 2 shows the training loss for $L=20$ and $L=60$. As expected, with low values of L the training is smooth and the cross-entropy loss is constantly reduced. Furthermore, regardless of the value of L, we can observe that the syllable loss starts higher than the note and duration one and it generally requires a higher amount of iterations to converge. This is due to the fact that the syllable space is much larger than the notes and duration ones. We also tried training with sequences equal to $L=500$, which is far slower in terms of convergence, as expected.

3.3 Generation

Once the network is trained, the generation process is iterative: we feed the network one pitch with the corresponding duration and syllable and it outputs the probability distribution for the next set of attributes. We then sample

[2] Name: *lmd-full_and_reddit_MIDI_dataset*

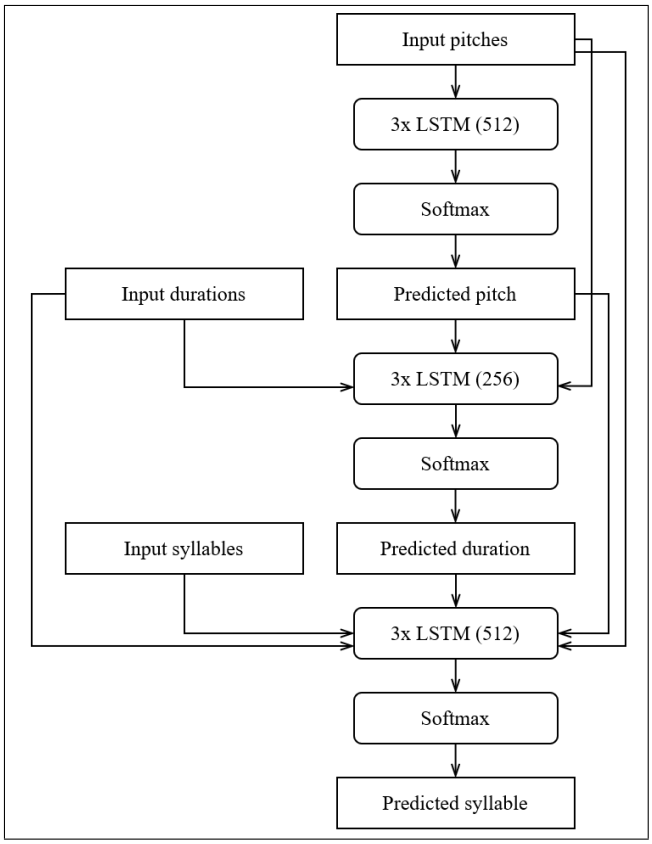

Figure 1. Overview of the neural network. The model first generates one pitch, then one duration conditioned on the sampled pitch, and finally one syllable conditioned on the sampled pitch and duration. The squared rectangles represent the input and output layers of the network whereas the rounded rectangles represent the trainable layers. Furthermore, the first number in every LSTM block refers to the number of layers and the second one indicates the amount of units in each layer. Finally, *Softmax* indicates a dense layer with a softmax activation function.

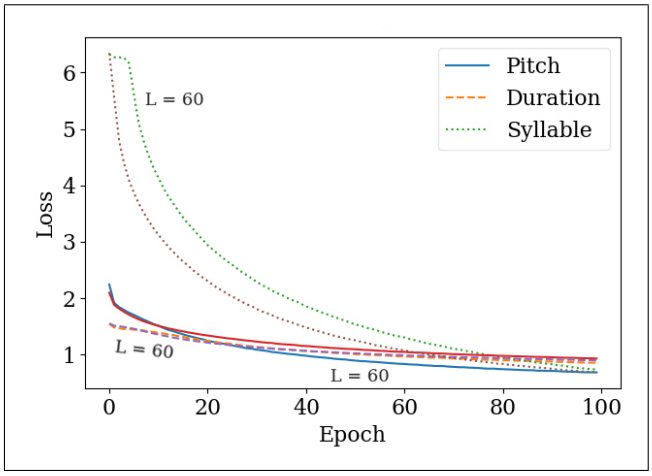

Figure 2. Training losses for $L=20$ and $L=60$.

from the probability distribution according to a temperature hyper-parameter T. Low temperature such as $T = 0$ is equivalent to selecting the token at the mode of the distribution. Opposed to this, higher temperature such as $T = 1$ increases the sensitivity towards low probabilities candidates, resulting in more diversity and "creativity". We set $T = 1$ when sampling the distribution of notes and durations and to 0 when sampling the distribution of syllables, resulting in a more creative melody with a more coherent and meaningful text. Once sampled, we give the attributes as input to the network for the next time step and the process is iterated until the desired sequence length is reached.

3.4 Evaluation metrics

We use several quantitative evaluation metrics to evaluate the performances of our systems:

- Pitch distribution: a count of which pitches occur throughout the sequences
- Duration distribution: a count of which durations occur throughout the sequences
- Interval distribution: a count of which intervals occur throughout the sequences
- N-gram distribution: a count of how many MIDI numbers N-grams repetitions occur throughout the sequences
- MIDI number span: the difference between the highest pitch and the lowest one
- Number of unique MIDI numbers: a count of how many unique pitches are present in the sequence
- Song length: the sum of all the durations in a sequence

We also evaluate how often the generation of consecutive syllables form sensible words. Let S be the generated syllable sequence, and denote by $|\hat{S}|$ the number of syllables in S that do not form real words with its sequence neighbors. Denote by $|SU|$ the unique set of S. The proportion of syllables that do not form a word is:

$$S_{norm} = \frac{|\hat{S}|}{|SU|} \quad (1)$$

4. RESULTS AND EVALUATION

We build networks modeling sequences of the three different lengths. At first we selected $L = 20$ and then we explored intermediate values between 20 and 60. Our preliminary experimental results showed similar outcomes for these lengths. We also decided to test the extreme case of $L = 500$. Once we trained the three networks, we used them to generate 500 samples each and then evaluated them using the metrics defined in section 3.4. In a similar fashion, we defined the baseline by computing the same metrics on the sequences of the training set.

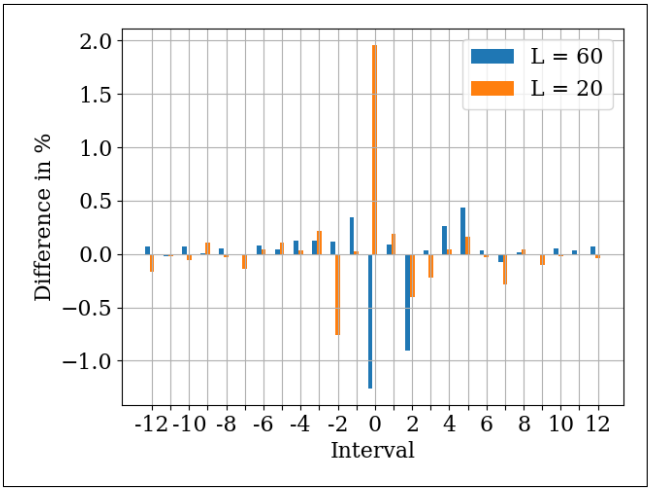

Figure 3. Difference in percentage between the note intervals distribution of the generated samples and the baseline for different sequence lengths L. For example, samples generated by the $L = 20$ model have about 2% more unison intervals than the training data sequences.

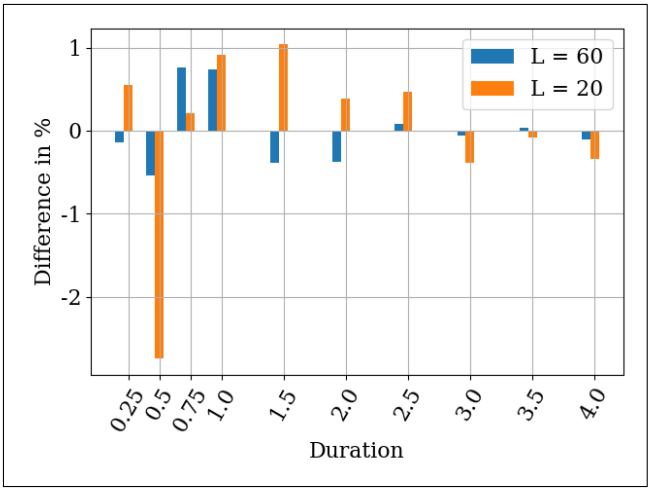

Figure 4. Difference in percentage between the note durations distribution of the generated samples and the baseline for different sequence lengths L. For example, samples generated by the $L = 20$ model have about 3% less quavers than the training data sequences.

The pitch distribution of the generated samples for the three L considered resembles the distributions of the baseline: most of the sampled pitches are in the range $[55, 81]$ and the network can ultimately predict a note with an accuracy of 69.85% for $L = 20$, 76.93% for $L = 60$ and 87.76% for $L = 500$. This can be further observed by looking at difference between the interval distribution of the generated samples and the baseline, depicted in Figure 3: for all the three considered sequence lengths, the distributions closely resembles the baseline, and perfect unison, major second and minor third appear as the most common intervals.

Likewise, as it can be seen from Figure 4, the duration distribution for $L = 20$, $L = 60$ and $L = 500$ results very similar to the baseline ones, indicating that the model

Model	L	MIDI number span	Number of unique MIDI numbers	Song length	S_{norm}
baseline	20	26.658	5.869	76.682	0.444
trained	20	26.836	5.822	75.862	0.645
baseline	60	31.406	8.344	229.831	0.540
trained	60	32.028	8.410	231.472	0.816
baseline	500	31.074	12.470	1931.7	2.244
trained	500	36.38	15.91	2011.57	67.54

Table 1. Number of training sequences and information loss for three different sequence length L.

successfully captured the transition probabilities between durations. Indeed, the network can predict a duration with an accuracy of 68.70% for $L = 20$, 69.52% for $L = 60$ and 70.62% for $L = 500$.

The N-grams distributions present generally lower values compared to the baseline, and this is especially true in the case of high N-grams values such as 4 and 5. This results in fewer repetitions of motifs and arpeggios in the generated sequences as well as less melodic coherence over long time spans.

Table 1 shows a summary of the numerical statistics. As it can be seen, the models trained with $L = 20$ and $L = 60$ very closely resemble the baseline, showing once more that the network is able to produce similar results to the training data. Furthermore, the S_{norm} value indicates that the network is able to successfully generate consecutive syllables that form meaningful words, indicating that the model learned coherent linguistic properties. In the case of $L = 500$ the trained model presents a higher variance in all the numerical metrics compared to the baseline and especially in the case of S_norm we can observe that the value is more than 30 times higher. This reflects on the accuracy of the syllables prediction: it is as low as 32.18%, indicating that the network in the case of fewer training data and longer sequences is not able to successfully capture the transition probabilities of syllables.

Figures 5 shows two examples of generated samples, converted to staff notation. More will be available on a dedicated website after the paper is reviewed. Supplementary material provides many more examples.

5. CONCLUSION AND FUTURE WORKS

In this paper, we presented a neural network capable of generating lyrics and melody concurrently. The proposed architecture extends the work of [3, 4] and uses a similar RNN model with the addition of conditioning on the syllables. The evaluation metrics validated the output sequences of the network, showing that the generated material in terms of pitches, durations, and intervals closely resembles the training data.

Nonetheless, several future works can further improve the performance of the network. One way could be to narrow down the problem: the dataset used spans over a wide variety of different genres and styles. With more data available, the network could be trained for one specific genre, and this would result in the generation of more coherent sequences. A more complex alternative would be to add a *style* input and output units. These could act as latent variables that could be used at generation time to bias the network toward the generation of melodies and lyrics in the corresponding style. Furthermore, we have been doing some preliminary work in gathering and applying the method to a Gregorian chant datasets. We believe that the intrinsic relation between music and text of this genre would make it an ideal application for our method. Finally, it would be interesting to compare our method to the ones that generate melody conditioned to the lyrics and vice versa, in order to investigate whether the concurrent generation can more accurately capture the relations between notes and text or not.

6. REFERENCES

[1] Lakh MIDI Dataset. `https://colinraffel.com/projects/lmd/`. Accessed: 2020-04-15.

[2] The largest MIDI collection on the internet. `https://www.reddit.com/r/datasets/comments/3akhxy/the_largest_midi_collection_on_the_internet/`. Accessed: 2020-04-15.

[3] Florian Colombo and Wulfram Gerstner. Bachprop: Learning to compose music in multiple styles. *CoRR*, abs/1802.05162, 2018.

[4] Florian Colombo, Alexander Seeholzer, and Wulfram Gerstner. Deep artificial composer: A creative neural network model for automated melody generation. pages 81–96, 03 2017.

[5] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and yihsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. 09 2017.

[6] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NIPS*, 2016.

[7] Eric Malmi, Pyry Takala, Hannu Toivonen, Tapani Raiko, and Aristides Gionis. Dopelearning: A computational approach to rap lyrics generation. *CoRR*, abs/1505.04771, 2015.

[8] Bob Sturm, João Santos, Oded Ben-Tal, and Iryna Korshunova. Music transcription modelling and composition using deep learning. 04 2016.

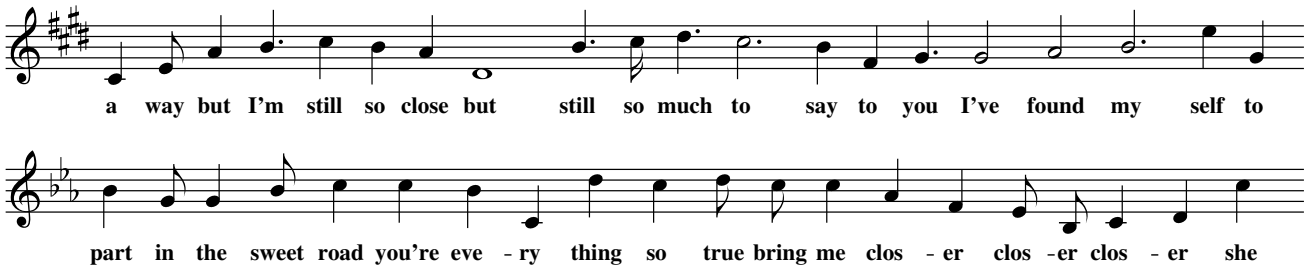

Figure 5. Two example outputs of the system generating sequences of length $L = 20$.

[9] Kento Watanabe, Yuichiroh Matsubayashi, Satoru Fukayama, Masataka Goto, Kentaro Inui, and Tomoyasu Nakano. A melody-conditioned lyrics language model. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 163–172, New Orleans, Louisiana, June 2018. Association for Computational Linguistics.

[10] Yi Biao Yu and Simon Canales. Conditional lstm-gan for melody generation from lyrics. *ArXiv*, abs/1908.05551, 2019.