FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

BACHELOR THESIS

Marek Židek

Generating polyphonic music using neural networks

Institute of Formal and Applied Linguistics

Supervisor of the bachelor thesis:	Mgr. Jan Hajič
Study programme:	Informatika
Study branch:	Obecná informatika

Prague 2017

Title: Generating polyphonic music using neural networks

Author: Marek Židek

Institute: Institute of Formal and Applied Linguistics

Supervisor: Mgr. Jan Hajič, Institute of Formal and Applied Linguistics

Abstract: The aim of this thesis is to explore new ways of generating unique polyphonic music using neural networks. Music generation, either in raw audio waveforms or discretely represented, is very interesting and under a heavy exploration in recent years. This thesis works with midi represented polyphonic classical music for piano as training data. We introduce the problem, show relevant neural network architectures and describe our numerous ideas, out of which one idea, our experiment with three versions of skip residual LSTM connections for music composition, we consider a good contribution to the field. In related work, skip-connections were explored mostly for classification tasks, however, our results show a solid improvement for music composition (e.g. 47% of respondents considered our samples real). We also show that skip-connections have rather diverse hyperparameter space for future tuning. Apart from standard automated test set evaluation, which is hard to design and interpret for creativity mimicking models, we also did a complex evaluation through surveys. The evaluation was specifically designed to not only to show results for our samples, but to reveal information about expectancy, preconceptions and influence of personal characteristics of the respondents. We consider this a valuable resource for future works in automated music generation.

Keywords: music generation, computational composition, recurrent neural networks, deep learning

I would like to thank my relatives, friends and my girlfriend Zuzana for mental support during tough times. I would also like to thank my respondents for their patience with my long surveys. My biggest thanks comes to my supervisor Jan Hajič jr. who didn't hesitate to spend time, taught me many new things from both technical and musical spheres and always stayed supportive.

Contents

Introduction

Automated generation of a work of art is a very attractive field, because it could enhance our look at art or help artists find interesting links to stimulate their creativity. It is also an interesting problem for artificial intelligence because of the complexity and abstractness of the musical patterns. We will use advanced deep learning[1]. methods because rigid algorithms imply that a programmer already has an idea on how to mimic creativity and, as we can see from recent deep learning results (AlphaGo [3], DeepStack [4]), neural networks might have an edge over human performance

In this work, we'll train Long Short-Term Memory (LSTM) networks [5] as music generators. Such a use means that we will approach our problem sequentially and train the network to be able to generate one step of our sequence based on a previous step and feed the generated step back into the network to generate the next step and so on. This means that we only have to provide a first step.

One problem with music generation is that it is hard to evaluate the models. The metric that is used for the training phase (cross-entropy/perplexity) is harder to interpret for validation purposes, because in generation and music generation particularly, we cannot only evaluate similarity. We will see an automated evaluation on the well known Nottingham dataset of folk music and then a qualitative and quantitative evaluation by surveys (with ratings in relevant categories and Turing tests to show if our results can fool) in comparison.

We experiment with various models and discuss our results, but most importantly, we describe and evaluate a new model for generating music. Our model uses residual skip-connections [6] through time with periodically weighted connections to force the network to capture longer time dependencies and increase coherence of the outputted music samples. To our knowledge, no other work used this approach before for generating music. We discuss and evaluate various ways of adding the skip-connections and show that our best model's resulting music has fooled 47% of respondents into thinking it was coming from a real composer, where the baseline fooled less (39%). We also show that the results from surveys are comparable to our automated evaluation on test set and to our own subjective opinions.

We will show a robust scheme for evaluation by 4 different surveys which doesn't only evaluate the models, but also reveals something about expectancy of the respondents and their preconceptions about algorithmic composition. It also shows possible weak and strong points of our models.
To implement models in this thesis, Theano [7], Tensorflow [8] and Keras [9] libraries were used along with Python 2.7 programming language.

[1]In this thesis, we assume the reader has a basic knowledge about machine learning and deep neural networks. For a good introduction into deep learning, see Goodfellow et. al: Deep learning, 2016 [1]. For a broader overview of work in Deep Learning up to 2015, including its historical roots, see Schmidhuber, 2015 [2]. Introduction to music theory and a brief glossary of music terminology is part of this work.

1. Problem Statement

In this chapter, we attempt to introduce the very basics of music theory and the corresponding terminology that is used throughout this work. We are also going to describe what is our problem and goal from machine learning perspective

1.1 Music as a creation

We will see that music has many rules, repetitions and routine, meaning it is well suitable for algorithms to compose it. However, the reality that makes music interesting or classified as good, is to stick to these rules and, from time to time, to purposefully and completely break them. More specifically, music can contain structural relationships between the dimensions of musical objects (we will see *pitch, duration, harmony, rhythm, timbre etc.*) but are also many non-mathematical elements including tension, expectancy and emotion. To capture these signs of creativity, machine learning is starting to be used and explored.

1.2 Music in general

Music can be described as a sequence of *notes* in time. Notes have four main characteristics: *pitch, duration, strength (velocity), and timbre.* When *notes* are performed, pitch means frequency, duration is the amount of time the note sounds, strength is the loudness of sound and timbre describes spectral characteristics and additional noise. Pitch and duration are the most important and create a base of music compositions. Strength and timbre are more complex from mathematical view. They are sometimes left for the musician to interpret with only little clues.

1.2.1 Notes

The musical objects we've seen in previous section are called *notes.* **Pitch** is a categorical variable with values from a subset of frequencies defined by scale and tuning. For western music, the scale of available pitches consists of semitones, using a tuning system where the next higher semitone is determined by the previous as $s_{i+1} = s_i \cdot 2^{f\frac{1}{12}}$.

Pitches are ordered linearly by their ascending frequency and arranged into a repeating pattern (i.e. C, C♯,D,D♯,...) of 12 notes called semitones. One repetition of this pattern is called an *octave.* Octaves are numbered from 1 to 8, in the English-speaking tradition. The number of octave is usually concatenated to the name of note (e.g. C5). For a base octave and a note, the same note in any other octave can be derived by stacking multiplication or division the note's frequency by 2 (e.g. midi tone 48 (C4) and 60 (C5), are both on step "C", because the C5 pitch frequency is exactly twice the C4 frequency. That also means they resound a perfect harmony). In *MIDI*[1] format, the pitches are numbered by

[1]*MIDI* is a technical standard and a format that allows instruments, computers and other devices to connect and communicate. It specifies a lot of musical information through *event* messages, we will use its pitch, and duration events

integers in ascending order by their frequency.

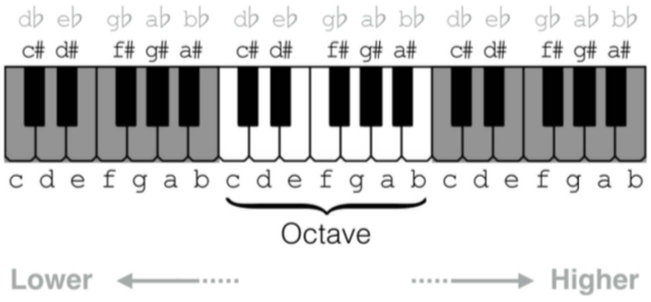

db eb gb ab bb db eb gb ab bb db eb gb ab bb
c# d# f# g# a# c# d# f# g# a# c# d# f# g# a#
c d e f g a b c d e f g a b c d e f g a b
Octave
Lower
Higher

Figure 1.1: Notes (named pitches), illustrated on a piano keyboard. For our purposes, we are going to consider these 12 semitones, even though musicians tend to express the same *frequency* with different *pitches* (e.g. C♯ and D♭), [10]

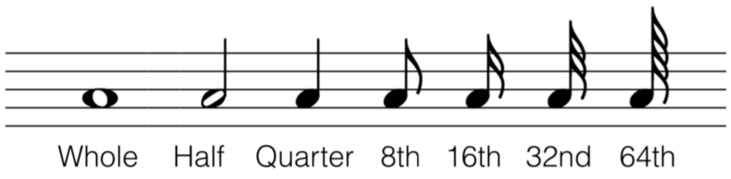


Figure 1.2: Types of notes according to duration. One whole note takes as long as two half notes, etc. The lowest duration we are going to consider in our models is the 16th. (i.e. we will omit lower duration notes in the training data), [10]

Duration is also categorical[2]. Duration is measured in *beats* (units of musical time). The duration categories are fractions of a beat, mostly in powers of 2. In musical terminology a 4-beat note is called a *whole* note, a 2-beat note is a *half* note, then *quarter*, *eight* etc. The most common grouping of beats into regular units (called *bars/measures*) is by 4, hence the name *"whole"*. However, durations that do not fit into this "powers-of-two" (e.g. 1/3 or 2/5) are not an exception in most of the compositions.

1.2.2 Intervals

Intervals are the differences, counted in semitones, between notes and can be easily found out by subtracting the two corresponding *midi* pitch numbers. The frequencies of tones can be calculated from an easy formula and derived from midi tone 69 (A4), that has 440Hz:

$$f = \sqrt[12]{2}^{(n-69)} \cdot 440Hz$$

.

How we, humans, perceive melody or harmony does not rely on concrete tones we hear, but mostly on ratios of their frequencies.

[2]categorical in music theory, although its realization is not: musicians slow down or speed up the flow of musical time in order to bring out inner structures in the music

Let us analyze the formula with common chords. For a tone that is one octave higher than any one considered, we can see that it is basically $\sqrt[12]{2}^{12}$, and that is exactly 2, so they create a perfectly consonant sound when played together.

Another important interval is so called the *third* (the name is explained in 1.2.3). The interval is 4 semitones long (+4) and is sometimes called a *major third* and we will call it like so. When placed into the formula, $\sqrt[12]{2}^{4} = 1.2599$ meaning that note e.g. E has almost exactly 25% higher frequency than C i.e. for 4 oscillations of C there are approximately 5 oscillations of E, so they consonantly resonate when played together.

An interval, which appears in almost every composition is the *fifth* (perfect fifth), where the interval is +7. $\sqrt[12]{2}^{7} = 1.4983$ i.e. for two oscillations of e.g. C, there are approximately three oscillations of G , which resonate even more consonantly than the *major third*, when played together. After the octave (+12), *major third*(+4) and *perfect fifth* (+7) are the most important intervals.

The whole concept of composition, if we over-generalize, is that there are, in certain 'waves', perfect harmonies alternated with less perfect, or even complete dissonances. There has to be a certain relationship between consonance and dissonance i.e. configurations of intervals sounding together. Otherwise, the listener would get bored. The theory [11] behind why it works is that consonance and dissonance are a function of centrifugal and centripetal[3] forces in music. We will use this theory as a reasoning for our design upgrades to neural networks. However, timing is really important as well, because this tension from the theory can be achieved through the use of timing alone (see e.g. composition for a solo drum).

1.2.3 Scales

A *scale* is a set of pitches (usually 7), which is sufficiently identified by a base tone and its type of harmony. Base to is just one of the 12 tones we know and type of harmony is "happy", "sad", "kind of cheerful but little out of tone", "with oriental feeling" etc. There are numerous different scales with different sizes and harmonies, we will focus, mainly because of historical reasons, on "happy" and "sad". They are called Major and Minor. They have 7 tones in intervals +0, +2, +4, +5, +7, +9, +11 for Major, +0, +2, +3, +5, +7, +8, +10 for Minor. Every tone in the scale sounds more or less consonant with the others from scale. A melody part of a composition is usually composed using a certain scale. The scale can be violated from time to time to create a moment of surprise, or can be changed completely multiple times throughout the composition.

Now, the names of the most important intervals the *third* and the *fifth* are much more clear as they are just orders in the two basic scales. Playing the base tone, third tone of a given scale (i.e. +4 in Major, +3 in Minor) and the fifth tone (+7 in both Major and Minor) is the most frequent chord in music and is called *triad*. It is the base for every harmony in our music.

[3] *"In Schoenberg's context, centrifugal forces are those that require expansion, like Phiolaus's unlimiteds, they constitute the potential for development with a musical idea. Centripetal forces are those that lead to coherence, they hold the idea together and make us perceive it as a unity."*, [11]

In this work, convolutional filters in GAN discriminator are designed to follow these rules and extract the features to create an abstract the concept of a chord, that can be freely transposed along note axis.

1.2.4 Rhythm

Unlike *tempo*, which is just a speed of a composition, rhythm is defining how the time axis is approached. Tones and harmonies from a certain scale, that are played randomly within the time are usually nothing pleasant. Music is usually split into parts of some fixed duration called bars or measures, which consists of any number of notes or rests, where the sum of their durations must equal the bar's size. The most used bar size is called the whole, or the 4/4, consisting of 4 quarter notes, therefore 4 *beats*.

In this thesis, rhythm is a very important part because the current position in a bar (not the bar position in a song) is used to help prediction, where the idea is borrowed from Johnson's [12]. This addition to the model, as the author see it, was crucial, because the outputted music finally started to have an idea about time. Music used as training data is, for these reasons, filtered by their rhythm.

1.3 Music generation

We will describe how a music composition can be encoded in computers, how we represent it to our models and what is our goal. There might be more "correct" representations and goals, because music offers a challenging array of representation problems, [13]. Therefore, in this section, we are going to describe the most general approaches or when we get closer to the specifics, the ones that was used in this work.

1.3.1 Representation

There are two main branches of representations of music. *Discrete*, which is close to standard music notation from historical Europe or *continuous*, which could be thought of as captured music performance through sound waveforms. While discrete representation offers the most important information for a performer, there are many ways how it can be performed (e.g. *strength, timbre, exact timings and duration etc.*). There are works and advances in machine learning algorithms for continuous representations, which we will see in Chapter Related Work 3. However, within this work, we will work with discrete representations, more specifically a *MIDI* representation.

While discrete representation comes in many forms, which are more or less applicable to machine learning (e.g. text based, applicable directly to char-RNN, [14]), midi (i.e. binary) files are better to be converted to a matrix, in our case $M \in \{0,1\}^{n \times m}$. Because of the sequential character of the music data, n represents time domain and can be of variable length. The time has to be quantified to reasonably small chunks that we perceive as one unit (e.g. duration of one 16th note). Pitch is represented by m (pitch is categorical) and together with time, 1/0 encodes "pitch m played/not played at time n". Duration is represented implicitly trough the amount of sequential played notes with the same pitch. This representation is the most basic and omits other characteristics of musical objects to simplify the problem, however, it is easy to understand. Important problem is that this representation doesn't differ between one note of a specific pitch played for n *beats* and n independent notes of then same pitch played for 1 beat. This can be fought [12] by adding a next $\{0,1\}^m$ dimension which represents articulation (makes notes independent). For different lengths of sequences, we need to either add padding or cut the lengths of sequences, omitting short ones.

Nevertheless, music, in our more specific case a discretely represented music, is challenging for representation itself and even more challenging for representation as a machine learning problem (i.e. defining input and output). Therefore, it is a subject of exploration where the most recent advances will be seen in Chapter Related Work 3.

1.3.2 Goal

Our goal is to generate a sequence considered to be a musical composition represented in a manner described, using a training data which is described the same way. We will use a sequential approach to both the training and generation. In this approach, we model a conditional probability. We know a part of the song at one time step n (**input** vector $\in \{0,1\}^m$), we also know all of the previous time steps we have seen and want to model probability of a next time step (*output* vector $\in \{0,1\}^m$). For reaching the goal, we can add additional information to input vector to emphasize a feature[4] that is readable from the representation, but hard to interpret for it's hidden meaning.

There is a problem that the dimension of the output vector is too big (e.g. 2^{88}) and such high dimensions are hard to handle with today's techniques, [15]. Therefore, the problem statement is (as also seen in Related Work 3) made simpler, but it is constraining a quality of the model. We model a conditional probability of m ordered Bernoulli random variables (in our case probability of pitch on/pitch off) independently, based on the input vector and historical context described in previous paragraph. The problem is with their independency because the individual note's pitches are dependent within one time step through *harmony* rules. We will see how to cope with this problem in related work and our ideas.

In our specific representation, we want to capture time dependencies between pitches, harmonies, disharmonies (dissonances), durations, articulations, rhythm (*beat* dependencies) and melody (dependencies on encapsulated subsets of pitches and durations). The next goals, which are of subjective manner to a person and are harder to describe to a machine learning model, are very important. These are tension, expectancy, surprise and idea within an outputted sequence.

[4] e.g. beat

2. Relevant neural network architectures

In this chapter, various general concepts from deep learning, that are used in this work are briefly explained.

2.1 RNN

RNNs-Recurrent Neural Networks are typically used for sequence processing and can be seen as a discrete-time system with an input x_t, an output y_t and a hidden state h_t, where $h_t = f_h(x_t, h_{t-1})$ and $y_t = f_o(h_t)$. This creates an environment suitable for modeling conditional time dependent probability. However, a major problem of this architecture are vanishing and exploding gradients [16]. There is an exponentiation term inside the gradient computation which can make the network either close to untrainable, or very unstable. From an equation that defines h in time t as $h^t = w^{(t)} \cdot h^{(0)}$, where $w^{(t)}$ denotes a weight matrix for t steps can be seen as $Q \cdot \Lambda^t \cdot Q^T$, where Λ contains n eigenvalues, that cause the gradients to vanish or explode according to being < 1 or > 1. (For more details refer to Goodfellow et. al [4].)

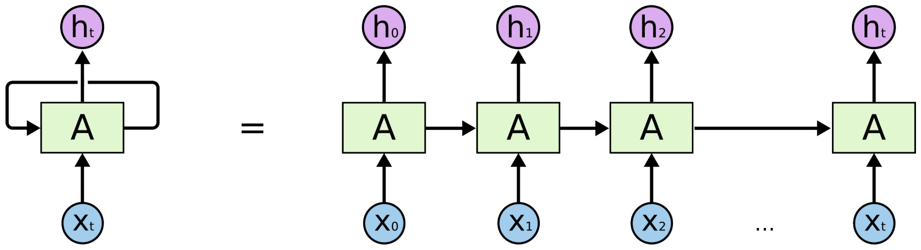


Figure 2.1: An unrolled recurrent neural network, [17]

Nevertheless, there are several advanced RNN architectures that surpass these problems e.g. LSTM or GRU variants. Another widely used concept is the Bidirectional RNN, where the future context is taken into a consideration as well while the output is computed.

2.1.1 LSTM

LSTMs-Long Short-Term Memory networks deal with vanishing gradient problem and can be successfully trained to process long sequences [18] [5]. It takes, just as RNNs, an input sequence $\boldsymbol{x} = (x_1, ..., x_T)$, hidden sequence $\boldsymbol{h} = (h_1, ..., h_T)$ and outputs a sequence $\boldsymbol{y} = (y_1, ..., y_T)$. The main addition is a gating mechanism consisting of i_t, f_t and o_t around a central memory cell c_t, that is built to capture long range context. Then, during forward propagation, network iterates the equations, where $t = 1, ..., T$:

$$y_t = W_{hy}h_t + b_y \tag{2.1}$$
$$h_t = o_t tanh(c_t) \tag{2.2}$$
$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{2.3}$$
$$c_t = f_t c_{t-1} + tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{2.4}$$
$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{2.5}$$
$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{2.6}$$

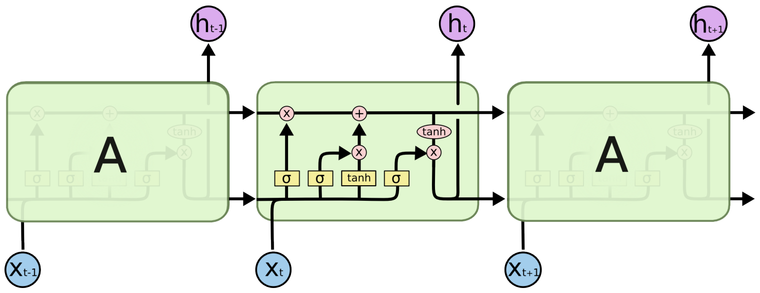


Figure 2.2: LSTM cell where the gradient flows lineary backwards through time steps, [17]

For a more thorough explanation, see Christopher Olah's blog [17], where different LSTM architectures are discussed and compared.

In this work, we use a variant LSTM-b of Josefowicz et. al [19], where 1 is added to the forget gate bias b_f for a better initialization, that doesn't have vanishing gradients early on. It is considered to be an improvement in almost any training scheme and some deep learning libraries such as Keras [9] set it as default.

2.1.2 Advanced LSTM variants

Residual connections, see He et. al (2015) [20] (1st place on the ILSVRC 2015 image classification task[1]), are shortcuts in very deep neural models, that help diminishing the vanishing gradient problem. Usually, for a better gradient flow, identity function is applied over a few trainable layers, added to the hidden layer and then activated. If dimensions are not the same, shortcuts are linearly projected to match them.

A layer with skip-connections computes the function:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + \boldsymbol{x} \tag{2.7}$$

or

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + W_s\boldsymbol{x}. \tag{2.8}$$

[1]`http://image-net.org/challenges/LSVRC/2015/`

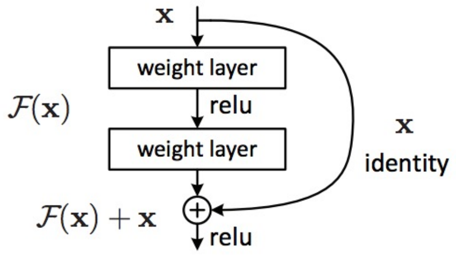


Figure 2.3: Residual identity connection, [20]

Skip residual connections

Concept of these shortcuts was, apart from deep convolutional models, shown to be a decent improvement for recurrent networks. Wang (2016) [6] or Kim et. al (2017) [21] show the effectiveness of such shortcuts. Highway LSTMs [22] have gradient shortcuts in internal cells, while skip-connected LSTMs (SC-LSTM) connects hidden/output layers. According to Kim et. al.: *"Each output layer at the residual LSTM network learns residual mapping not learnable from highway path. Therefore, each new layer does not need to waste time or resource to generate similar outputs from prior layers"* [21] *pg. 1.*

Wang [6] found that adding skip connections in a straightforward way, i.e. skipping two hidden layers, compromise the merit of gates that are already sufficient at maintaining short-range memory, and makes the unregulated information flow too strong. To enhance the LSTM information flow, skip-connections are added with a wide range (e.g. 20, or in this thesis 16 (size of music 4/4 bar) and 48).

$$L = 20, \forall t \in \{1, 1+L, 1+2L, ..., \left\lfloor \frac{T-L-1}{L} \right\rfloor\} \tag{2.9}$$

$$h_{t+L} = tanh(c_{t+L}) \otimes o_{t+L} + \alpha h_t \tag{2.10}$$

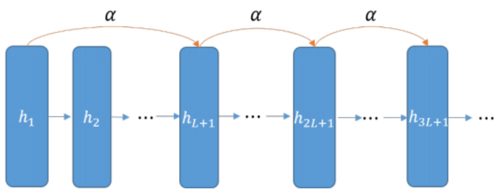


Figure 2.4: Wang et. al SC-LSTM, [6]

In this work, skip connections are investigated in more ways such as skip-dense i.e. added for all $t = 1, .., T$ because Wang doesn't report experimentation with frequency, or skip long range i.e. added with wider range of $L = 48$, thus connecting across two bars.

A way to regularize the strength of the information flow is through the α scalar in $h_{t+L} = tanh(c_{t+L}) \otimes o_{t+L} + \alpha h_t$. In our music generation experiments, skip-dense connections add to the overall integrity of a part, however, this strong consistency can drive the listener bored. Stronger information about the previous context of the song might be needed at the beginning of a bar, where through the bar, network can learn to improvise more and more.

Scheme described in this work for our models is $\forall t$ where

$$\alpha_t = \frac{1}{t \mod L} \tag{2.11}$$

$$h_{t+1} = tanh(c_{t+1}) \otimes o_{t+1} + \alpha_t h_t \tag{2.12}$$

as opposed to Wang's $\alpha = 0$ for all $i \notin \{1, 1+L, 1+2L, ...\}$.

Another way of adding residual connections, that seems more robust, is explained in a very recent paper Kim et. al (15 Mar 2017) [21], where the implementation and the explanation is more complex, and yet to be analyzed for music composition in future work.

2.2 CNN

Convolutional neural networks have their most common use in image recognition and they feature the use of mathematical convolution in a form

$$S(i,j) = (I * K)(i,j) = \sum_{m,n} I(m,n)K(i-m, j-n) = \sum_{m,n} I(i-m, j-n)K(m,n) \tag{2.13}$$

where $(I * K) \in \mathbb{R}^{m \times n}$. Convolutions have their trainable weights shared across a layer and work as feature extractors, where the feature dimensions are called channels. Stride s is the length of a step for convolutional windows. As the number of features grows, there is a need for downsampling, i.e. max pooling or average pooling. Pooling also provides a basic local translation invariance, helps to counter over-fitting and reduces computational cost.

$$maxpool(I)(i,j,k) = \max_{m,n} I(s \cdot i + m, s \cdot j + n, k) \tag{2.14}$$

where k is the output channel.

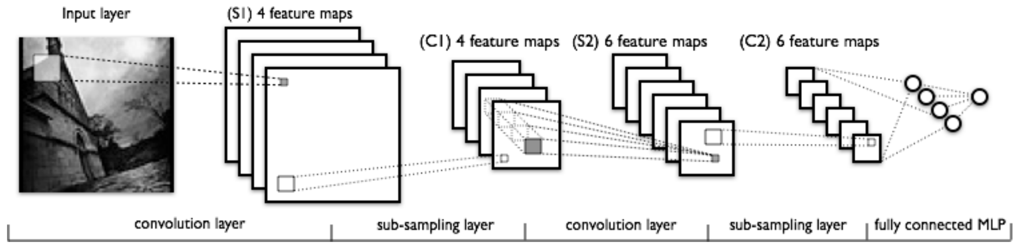


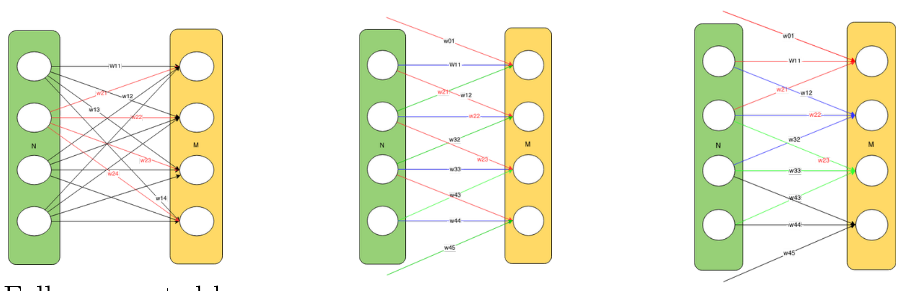
Figure 2.5: Example of a CNN architecture, *hyperverge.co* [23]

2.2.1 1D Convolutions

To process sequences, it is natural that the convolutional window has only one dimension and all the concepts from 2D convolutions are generalized. It is also possible, in multidimensional sequences, to perceive the feature dimension which in this work encodes the notes played simultaneously, as an input dimension and process the sequence as an image using 2D convolutions.

2.2.2 Locally connected layers

Locally connected layers are a hybrid between convolutions and dense/fully-connected layers. It is, in many ways analogous to convolutions, however, the weights are not shared. Where fully connected layers are limited by too large input and output vectors making the large weight matrix overfill computational memory, and where convolutions are too constraining, locally connected layers are a formidable help. Their weights matrices are still much larger then the ones of the convolutions, but not in a futile way. Taigman et. al (2014) [24] successfully utilized them in DeepFace, closing the gap to human-level face verification.



(a) Fully-connected layer (b) Convolutional layer (c) Locally connected layer

Figure 2.6: Locally connected layers compared to dense/fully-connected and convolutional layers, [25]

In this thesis, locally connected layers are used within music bars to capture time dependent information, and a context of surrounding time steps, where weights are unshared within a musical bar, but shared within beats across all bars of the song using time-distributed [9] wrapper layer.

2.2.3 Dilated convolutions

Dilated convolutions were investigated in Yu (2016)[26] or Zhou et. al (2015) [27]. Using the notation from [26], convolution with dilation factor l (where $l = 1$ is the standard convolution), is defined as

$$(F *_l k)(\boldsymbol{p}) = \sum_{s+l\boldsymbol{t}=p} F(\boldsymbol{s})k(\boldsymbol{t})$$

These networks work as multi-scale context aggregators and were successfully used in Google's Wavenet [28].

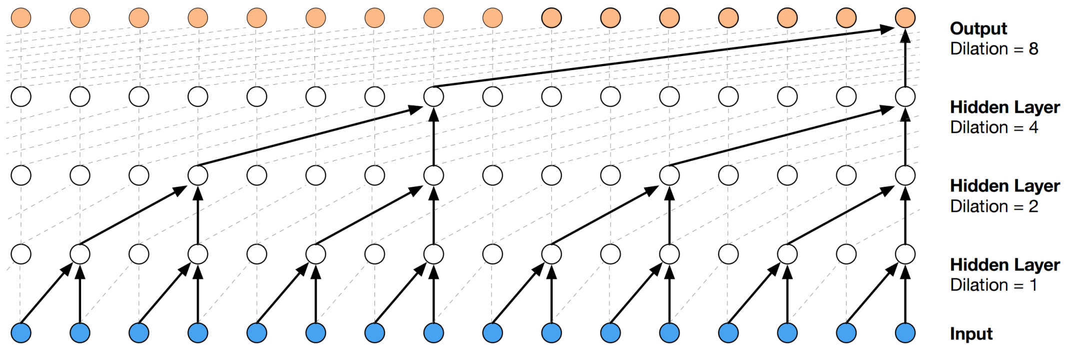


Figure 2.7: A stack of a dilated convolutional layers, [6]

In this thesis, they are used to capture the structure of notes preceding the currently evaluated time step with a time span of one musical bar/measure.

2.3 GAN

Generative Adversarial Networks (GANs) are unsupervised models that try to generate data from a distribution, given an uniform noise. It is conceptually a zero-sum game between two neural networks called generator and discriminator. The discriminator attempts to distinguish examples from real data from samples from the generator, and the generator is trying to fool the discriminator. Using notation from Goodfellow et. al (2014) [29], where they were first introduced, D and G play a minimax game with a value function:

$$min_G max_D V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(\boldsymbol{x})] + \mathbb{E}_{x\sim p_*(z)}[\log 1 - D(G(\boldsymbol{z}))] \quad (2.15)$$

A thorough explanation with proofs of convergence can be seen in [29]. The two models are joined together, first the generator, than the discriminator so the gradients can pass back from D to G. D has a single output neuron, with a sigmoid output function, that tries to distinguish 'real' data - from p_{data} and 'fake' data - from generator, that we try to model. The G, while D has its weights fixed, tries to update its weights using D gradients so that D would evaluate Gs output as 'real'.

2.4 Wasserstein GAN

Stacking two models together can yield very deep models, and even though D is usually implemented with LeakyReLU [30] activation functions, gradients passed to G are still a subject of vanishing gradient problem. LeakyReLU, or any of the more advanced rectifiers, are intended for better gradient propagation and to surpass the "dead neuron problem" [31] of ReLUs, where for various reasons (e.g. high learning rates) during the training, a neuron can return only one value and cannot recover from this state. Wasserstein GAN, from now on WGAN, introduced by Arjovsky et al. (1.2017)[32] deals with vanishing gradients in GANs using a different distance/metric. Where a standard GAN uses a MLE that in limit converges to the KL-divergence (technically not a metric) [33] and a sigmoid output function that allows for vanishing gradients, WGAN uses no output activation function and computes loss using Earth Mover (EM) distance. EM (Wasserstein [32]) distance between P_r and P_g is defined as:

$$W(P_r, P_g) = \inf_{\gamma \in \prod(P_r,P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||] \quad (2.16)$$

A complete understanding of this distance is beyond the scope of this work, but a very good blog post[2] explaining this is at [34]. There is shown, that there exist sequences of distributions that don't converge under 4 discussed divergences, but which do converge under the EM distance, and that for those 4 discussed

[2]http://www.alexirpan.com/2017/02/22/wasserstein-gan.html,2017

divergences, there are cases where the gradient is always 0. For an intuitive comprehension of the WGAN, D, rather than saying: ”I am sure that this is a real face”, says ”This face looks 3 times more real than the previous ones you've shown me” and than passes a very stable gradient to the G.

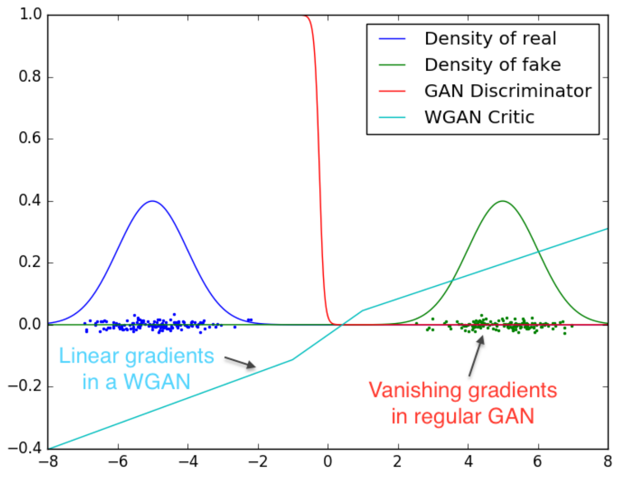


Figure 2.8: *”Optimal discriminator and critic when learning to differentiate two Gaussians. As we can see, the traditional GAN discriminator saturates and results in vanishing gradients. Our WGAN critic provides very clean gradients on all parts of the space.”*, [32]

We worked with WGAN because of the problems that regular GAN has were crucial for the implemented model to work, just as Agarwala et. al [35]*p.6* shows in applications of GAN to music generation, G keeps loosing to D. In our findings, regular GAN was trained to distinguish real music from generated, and early on, was found to label the data correctly, such as 99.99% real, or 0.01% real. Sigmoid function outputting these values have the gradients very close to 0 and make it hard for the G successfully use them to improve itself. WGAN showed to be an improvement, and because of all the theory described in [32] needs the weights to be clamped in the interval $(-0.01, 0.01)$ to work, this is also a good regularization for D .

3. Related Work

In this chapter, we are going to look at approaches to the problem of generating music. Recently, there has been increasing number of works trying to generate music. We are going to point out the most ones that influenced us the most.

We described our problem in Chapter Problem statement 1.3. The approach described is not the only one. We can see other works trying to generate music from a different perspective using a direct waveforms approach, such as Wavenet [28] using convoltional networks or GRUV using recurrent networks [36].

From works using discrete representation, we can see a work by Zimmerman [37], that tries to distinguish between a melody and harmony and model them independently[1]. In the same work, we can see an interval/differences approach to the problem of predicting a melody. The input and output vectors do not encode an absolute pitches but a difference in pitch relative to the previous time step. As this simplifies a problem by a large constant (all the possible pitches) but keeps the main idea of the composition, we also tried to investigate this approach too in 4.2. However, we applied it for a polyphonic scale where it is not very clear how to proceed.

Another work using discrete representation, which is very recent, Magenta's performance-rnn [38] is using a completely different representation from the one we explained in Problem Statement 1.3. It avoids the problem with high dimension (1.3.2) of the output vector by making the x-axis of the matrix representing music an arbitrary one, only encoding the order of events and not a specific time. The time is represented by events called *timeshifts* which encode a specific amount of time to be skipped while playing/not playing a described set of notes. The set of notes to be played is encoded by *on* and *off* events. All of the events[2] are encoded by one-hot vectors reaching much more reasonable dimension of the predicted output.

For our representation and approach with RNNs, the influental works were Eck et. al [39], Johnson [12], Boulanger-Lewandowski [40] or Google Magenta team [41] which all produce solid results. They workaround the problem with high dimensionality of the output vector by modeling each pitch independently, however, they provide methods how to cope with the lost dependencies.

The Johnson's [12] work is using a RNN that has recurrent connections not only for time axis, but also for note axis. The neural network has two recurrent layers for coping with time dependencies and the resulting output (in whole model, hidden) layer is passed to another two recurrent layers which are coping with note dependencies i.e. it models chords. Johnson's model is also different from our problem representation, because it doesn't take a vector of all notes from previous time step and output a vector of all notes that are predicted for current time step, but takes a note and its reasonably small context and outputs a probability for a note to be played.

The Boulanger-Lewandowski's [40] work is using an RBM (Restricted Boltzmann Machine)/NADE(Neural Autoregressive Distribution Estimator) for coping with note dependencies. For every time step, firstly an RNN copes with time de-

[1]only a current time step, not the whole time lines

[2]more events are described in, [38]

pendencies and RBM outputs reasonable musical chords. RBMs have a visible and hidden layer which models joint probability of visible vector and a hidden vector. In the RNN-RBM, for every time step a hidden recurrent state is passed to next RNN cell, and to both visible and hidden layers of RBM. For every time step, output from visible layer of RBM is passed to recurrent cell in the same time step. All of the connections are trainable. This work is considered very influential, however, we only used the representation of a problem presented.

Throughout this work, we re-mention the related work in more specific manners, when we describe our experiments.

From the related work, we can see that the representation and a goal we described in 1.3 is a reasonable approach, however, it has it's imperfections. The main problems that we face are joint distribution, mistakes (out-of-tune notes), note axis invariance[3], and coherence.

We try to add skip residual connections known from deep convolutional models, which we described in Section 2.1.2. The related work which used these skip-connections was also mentioned. However, no other work to our knowledge investigated the use of skip-connections for a music generation problem. We add to the field by experimenting and evaluating our 3 possible approaches to adding skip-connections.

[3]addressed by a very recent Johnson's [42] Tied Parallel Networks

4. The model

In this chapter, we will describe our approaches to music generation. In the first section, we talk about an LSTM generator and some tricks used for improving its outputs. In the second section, we outline a generative adversarial network (GAN) operating over the LSTM outputs that is designed to address some of the shortcomings of the generated outputs.

4.1 LSTM generation

This section describes the ways how LSTM was used for the generation i.e. predicting the set of played notes based on previous played notes and a historical context of the music part. We didn't experiment with GRU [43] cells, because it was shown in [35] that LSTMs are better fit for our task, however, the differences between the two advanced RNNs are very low [43].

4.1.1 First steps

For our purposes, a music part would be a binary matrix, where columns are representing reasonably small time steps and rows are representing note pitches.

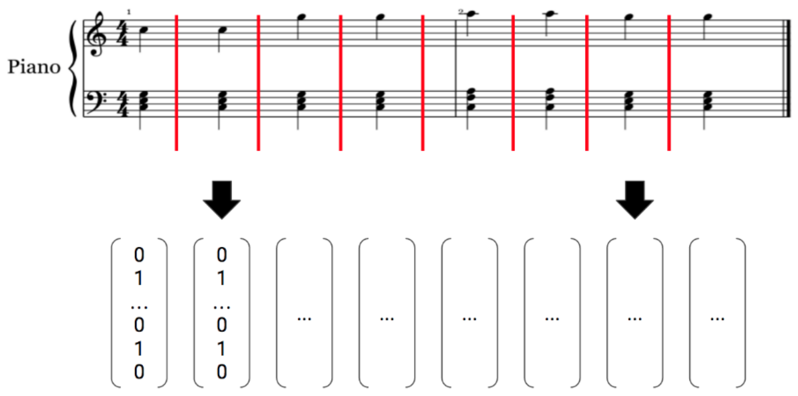


Figure 4.1: Visual interpretation of the basic representation, [37]

All music data is of sequential character and recurrent neural networks are well known for processing sequences, so they are always going to be at the center of attention for music generation. The approach we will use is to look at the current time step vector and have some memory, where the information about previous time steps is stored and then try to predict a vector of played or held notes in next time step.

The main goals of this project were to have the resulting music have some structure, to know when the part starts, ends, what main theme does it have or that it is good to sometimes play the same sequences of notes in key variations, to add disharmonies to not bore the listener. In a goal to predict a configuration of notes played at one time step based on the configuration of notes played on the previous, one could use standard feedforward networks but they could not take the whole song structure into account when predicting. When using LSTM, this concept is put on record by the use of hidden states and a cell memory, that keeps an idea about the song the entire time, not only one time step back. The memory is updated every time step, some information is added and some is removed. The network takes a binary tensor with the first dimension representing time, the second represents note pitches, and the third represents whether to play the note, and whether to prolong the note(i.e. ligature) or rearticulate, given it is already played.

4.1.2 Interval generation exploration

Firstly, with an urge for an innovative concept, which it was untill the release of new tied-parralell RNN [42] by Johnson, we would like the network to catch the main bluilding blocks, the intervals between notes at the current time (chords) and between past and now (melody). Then the music can be transposed along the note axis and still sound equally good, however, with a different ”atmosphere”[1]. We've seen in [37], that a simple idea about predicting the intervals in monophonic melody works well. To generalize this idea into polyphonic music, there are many options. The one that was explored is for the network to take a vector of relative differences between previous note and the current (counted from the middle of the vector). Now we would need a relative note value to which then interpret the next predicted output. The idea of taking a mean of notes is clearly bad, because in music, unlike in e.g. image generation, even a semitone out causes disharmony. The highest pitch seems to be a reasonable choice, because it often creates the melody part of the composition and that is a part the listener usually cares about the most. The bass line is also very important, especially for harmony, however, we think that the experiments would yield similar results.

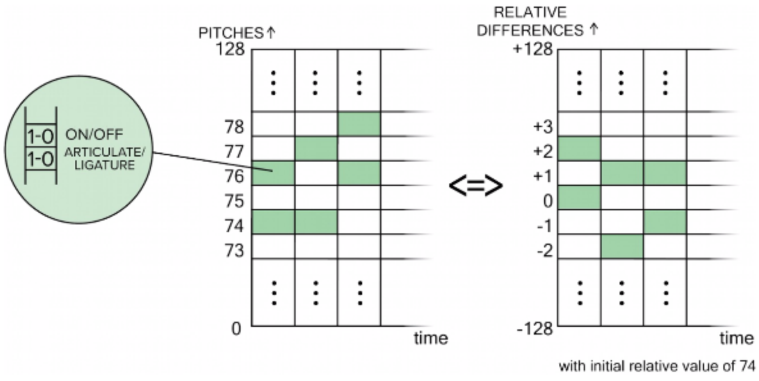


Figure 4.2: Visual explanation of the difference matrix

However, the LSTM could not capture the history of adding too much higher pitch notes (in a sense relative to the the previous note) in a row, resp. lower pitch notes in a row. That means, for a few seconds, the music has very good harmonies and melodies, but only on a scale of a few time steps, but as a whole it jumps across the pitches and eventually goes too high or too low and then the song gets out of the midi pitch boundaries. Therefore, we skipped these experiments for more promising model.

[1]which, as the author believes, is mostly arbitrary

4.1.3 Look at the LSTM model

After unsuccessful experimentation with the interval training, we rather made the LSTM take the vectors representing direct/absolute pitches of notes, where a binary representation of beat is concatenated to that input vector. Beat is there, because without it the compositions lacked a rhythmic structure.

This model, along with it's improvements up to and including Section 4.3 is the one we documented in attached programmer/user documentation[2] and "polished" our code for presentation.

Other important characteristics of music like dynamics, tempo or expressive timing were omitted for their high complexity and lack of related work. However, a very recently (29.6.2017) published work by Google Magenta [38], which we also pointed out in Related Work chapter, impressively experiments with these characteristics, which add to the feel that not only human-like part was composed, but also performed in such a way.

The input and output layers of our network are binary vectors of length 160 and 156 respectively. The span of notes is 78, that is because it is cut off of too high or low pitches from the 128 pitches midi representation, that almost never appear in compositions. This span is doubled due to the need to not only determine which pitches should be played, but also check that if a given pitch is played in a previous time step, whether to re articulate or prolong the duration of a note. For the input layer, the beat indicator takes 4 bits, because the quantization length is 1/16, and then the beat just repeats in cycles.

The cost function is a binary cross-entropy defined as

$$-\frac{1}{N}\sum_{n=1}^{N}[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)] \tag{4.1}$$

applied independently for each dimension of the output. It is usually used after a sigmoid output activation. This cost function, along with a categorical cross-entropy, usually applied after a softmax output activation, which models probability distribution, is most common when we try to decide the right outcome by argmax or use the modeled distribution to sample from it. Note that during the training phase, the ligature/arcticulate probabilities are masked by the binary mask of played notes to forbid learning the ligatures, where they cannot be played.

In the generation process, random sampling from the predicted probabilities is used, rather than thresholding. In such a way, there is still a posibility for random and unlikely surprise tone developing musical idea which can enhance listeners experience. When generating the samples, the first time step comes from the training data and all the other time steps are the outputs from the previous ones, which are fed back to the LSTM. This way, it can generate unlimitedly long.

The network has 3 stacked LSTM layers with the number of neurons 300, 300 and 250. We used initialization of forget gate's bias to 1 as it was shown to be an improvement in almost any task [19]. Dropout 7.1.2 can be used to counter overfitting, however, lower values are recommended because the network than has struggles with generating more diverse outputs. We used 0 and 0.2 dropouts, as

[2]documentation - `Program_documentation.pdf`, program - `music_network` for further navigation

the 0.5 was harder to train in our conditions, producing unsatisfactory results. To reduce the lengths of evaluation surveys, we used only stable 0 dropout, unlike in automated evaluation, where we tested both variants.

The on-line training (i.e. `batchsize = 1` method) argumented in [44] was used over minibatch training, as this worked better with lower number of data and it countered overfitting. Although batch training uses better estimation of the true gradient, empirical results on 27 different machine learning tasks showed that on-line training is faster and with no apparent difference in accuracy [44].

4.2 LSTM model improvements

The resulting music will be covered in the **Results** chapter, but we will see that the generated music has very good structure and an idea of time and chords. One problem is with the sudden out of the tone notes, however, that is not that much frequent to bee thought as an intention. Another problem, that is very hard to see at first glance, are the long term dependencies. An inexperienced listener might not see that, because the melody and structure fits together within a time long enough to not recognize, but the real world classical compositions feature repetition and variation (e.g. in modern music, the chorus) over a long time. Repetition in a short time period can bore the listener, but over longer time periods can make a feeling of looking forward to it (in case the song is known) or to make a feeling that the listener heard this before and liked it.

4.2.1 Majority voting during generation

As our model uses sigmoid output for every note independently, mainly because of the $2^{88}+$ configurations, the network has less of an idea what it is going to output. The following idea tries to deal with multidimensional dependencies, which the model has a hard time capturing. Apart from ensembling, where many models are trained parallely, here we use only one trained model and as the generating process uses randomness, we can get different results.

A simple illustrative example would be two chords, both containing e.g. 3 notes, that fit the next time step equally well, but the generated output (a sample) from the modeled distribution can play some notes from both chords which do not resound the wanted harmony.

A simple majority voting method is used, where the number of voters is 5 and their number of votes is not explicitly conditioned (only by randomness and a random vector of Bernoulli variables). It cannot be used efficiently to counter the discussed example, where both chords have same probability, no matter how unlikely/likely this is to happen. However, it can change the resulting probabilities in a way of increasing a probability of a likely harmonic outcome and lowering the probability of mistakes.
For the voting to work, we need a boundary for the number of notes played simultaneously in a given time step. It could be done by simply averaging the number of votes. The idea would work even without the use of bounds or votes, through numerical methods. However, both these approaches just work cooperatively with the main model. To increase the overall model complexity, we use another recurrent network. It gets exactly the same input as the previous one

and models a regression of the number of notes played simultaneously in a given time step. This model can be and was trained independently. It can capture **more** of the important dependencies because of it's focus on a simpler task.

Another aspect of this method is that it, instead of (e.g. tresholding) trying to eliminate the mistakes completely, tries to reduce their chance, but keep their possibility as an occasional exclusion from a pattern can help develop the musical idea. Dealing with it in the following local context could be the main interesting part of the resulting music samples.

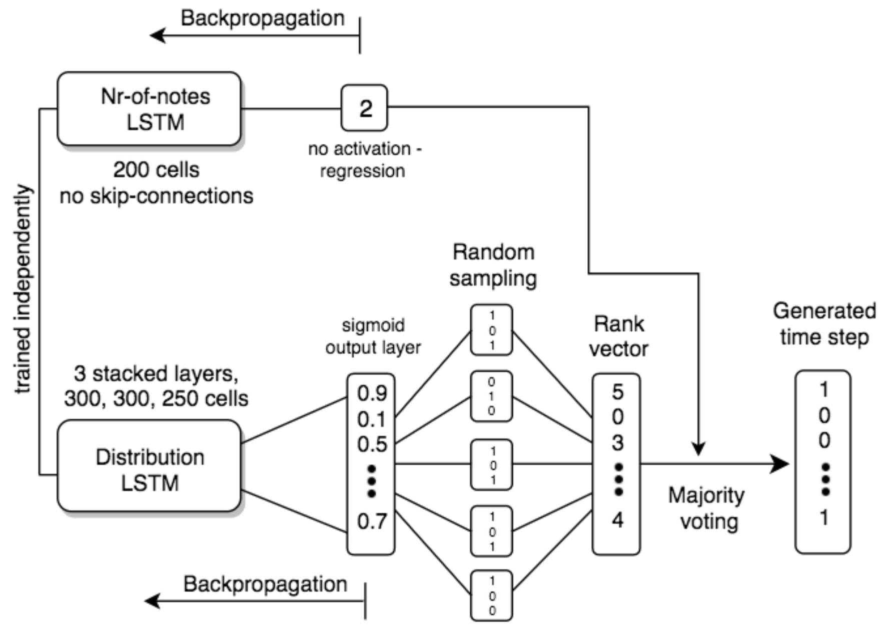


Figure 4.3: Interpretation of the generative process we describe for making the output more coherent.

The so called *nr-of-notes* model is also an LSTM with one hidden layer of 200 neurons and a ReLU output activation, predicting regression of one number with MSE error $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}(\hat{Y}_i - Y_i)^2$.

Using this concept helps the model to deal with errors coming from playing the notes with a very low probability, e.g. around 5%. Even though, in general 5% is considered a very low probability, it is accumulated in time steps, so the boosted model produces cleaner compositions. We use the explained the method for all the models we evaluated.

4.2.2 Experimental details

This section describes details and small improvements to the resulting music.

- The outputted probabilities for the ligature(prolong)/articulate, from the main LSTM models was augmented by exponentiation by an empirically found constant of 1.1 in favor of ligatures.
- During the majority voting method, it makes little sense to vote for the ligatures and predict their number, given that the actual played notes are not decided yet. Because of that, the ligature choices are thresholded,

where 2 and more votes means an articulation. Interesting insight is that the ligatures seem to have a high entropy as changing the empirically found constant by 1 point can produce very different results[3].

- If no vote was cast (given the randomness and low probabilities), we play nothing that time step, even if the other LSTM network for the number of notes predicted something else.
- During the majority vote, when the amount of notes left to play is less than the number of candidates with the same number of votes, we sample the needed amount from uniform distribution.
- To counter silent moments in compositions, constant of 0.3 was added to the output of the LSTM predicting the number of notes played simultaneously, adding more than this constant can seriously affect the ability of the model to play pauses.

4.3 Skip-connections

To help the model cope with longer time dependencies, skip residual connections discussed in Section 2.1.2 can be added. Hyperparameters of these connections would be their length, frequency and weights. In this work, we consider three models with skip-connections, all slightly different from [6]. We evaluate all of them on well known Nottingham dataset, and the subjectively best model against a base LSTM model by surveys, on a more interesting dataset.

- In Wang et al., [6], the connections are of length L and with frequency of one per L time steps. Wang states that adding skip-connections with a low L value is compromising the merits of LSTM gating mechanisms. However, there was no explanation or experimentation with the frequency of skip-connections. Our *skip-dense lstm* is a model that adds $L = 16$ long connections with a frequency of one per every time step. However, we found that the amount of additional information to the memory cell leaves little potential for a development within a musical idea. Resulting music is very coherent and boring to listen to. See `bad_examples/skip-dense.mid`.
- Inspired by the idea of temperature from simulated annealing [45] (Sec. 10.12), we could increase or decrease the ratio of expectation and surprise in music when it is most needed. The *skip-temperature lstm* tries to do that, however, the part of determining the needed moment is done arbitrary, by the structure of musical bars. It connects, just like *skip-dense*, $L = 16$ long connections with one per every time step, but the information flow through the connections is constrained by temperature (weight α_t), which grows and drops periodically $\forall t$ where

$$\alpha_t = \frac{1}{t \mod L}, \tag{4.2}$$

[3]see `bad_examples/articulation-favor.mid`

$h_{t+1} = tanh(c_{t+1}) \otimes o_{t+1} + \alpha_t h_t$. To the best of our knowledge, this residual connection weighing scheme has not been previously attempted, at least for generating music.

- *Skip-longrange lstm*, works just as the previous variant with the periodically weighted skip connections, however, the length of the connections is tripled to $L = 48$. We tripled the length of the connections to better connect the passages of a composition carrying a fresh musical idea to be coherent with the rest of the music. The length of the previous version connections was too low as the $L = 16$ (the length of one musical bar/measure) connects only little parts of the essence we wanted to capture.

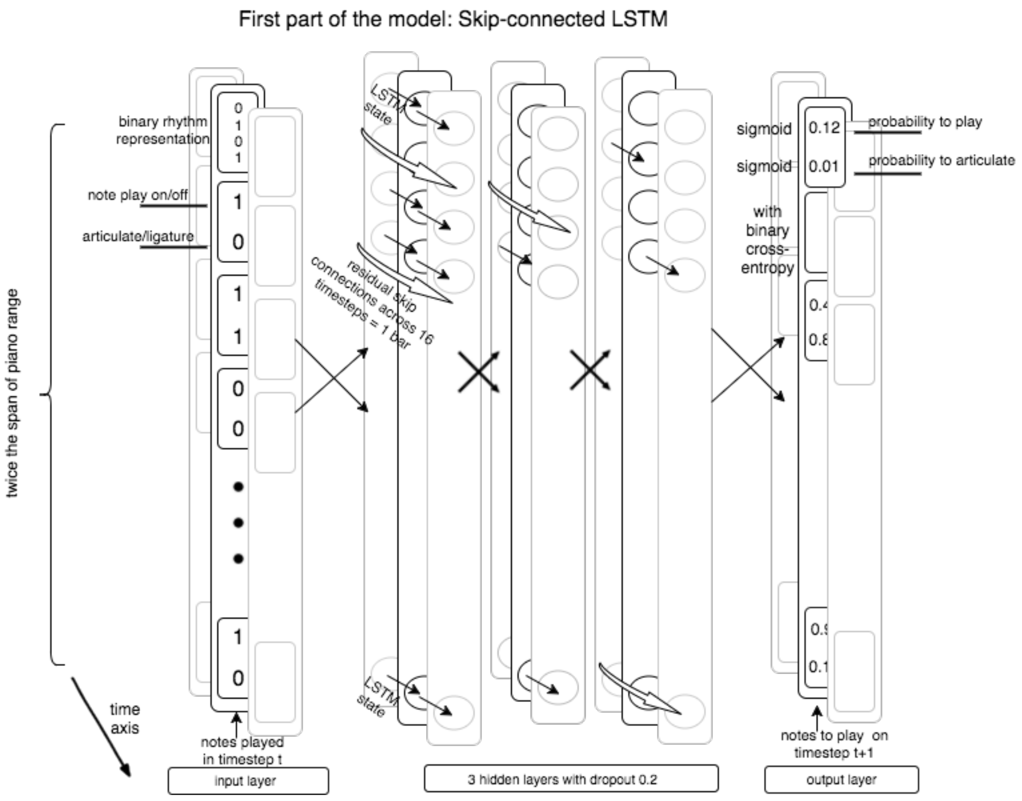

First part of the model: Skip-connected LSTM
binary rhythm representation
note play on/off
articulate/ligature
twice the span of piano range
time axis
notes played in timestep t
input layer
LSTM state
residual skip connections across 16 timesteps = 1 bar
3 hidden layers with dropout 0.2
sigmoid
sigmoid
with binary cross-entropy
probability to play
probability to articulate
notes to play on timestep t+1
output layer

Figure 4.4: Visual interpretation of the LSTM skip-connected architecture we use in this thesis

4.3.1 More interval generation exploration

Using these enhanced models and feeding them with relative interval difference matrix still proved futile, so a new idea of penalties was investigated. The idea was to keep an eye on the highest pitches, recalculate them with every time step and add a penalty to the cross-entropy loss based on the MSE loss between correct highest pitch and predicted highest pitch. That would keep the advantages of interval training and keep the network from producing music out of the midi boundaries. However, two attempts to make use of this idea found a dead end, mainly because of the scope of their correct implementation.

The first, that is, using a functional theano API for an algorithm with a deterministic way of finding out current highest pitch, that allows gradients to flow through it (it being a part of symbolic computational graph), was found to be a huge addition to the training time (Approx. 20 - 40 times slower).

The second, which is using a standard feedforward network to predict the highest pitch based on previous highest pitch and a new interval vector(definitely faster than the first method), was found to be a hard concept for the network to generalize. Even to train the network just to predict the first non-zero index using algorithm generated data with a uniform distribution of the 'first non-zero index' to increase the information gain of the dataset, was found to be harder than expected and left for the future work.

4.4 Incorporating harmonic constraints

We transformed a problem where the output is a maximum likelihood estimation over 2^{88} possible options to 88 independent estimations of a Bernoulli random variables. However, this completely ignores the fact that they are highly dependent, exactly because of the intervallic relationships described as harmony. Lewandowski et. al [40] deal with joint distribution of the output layer with the use of the restricted Boltzmann machine and Johnson [12] with a biaxial RNN, where the "bi" stands for a second note axis input into RNN architecture and both of the methods shows a good performance. In our attempts, convolutions were used on the output layer. These convolutions are applied within a GAN architecture to correct the mistakes of the naive approach by addition, deletion or modification. We tried a process that is taken step by step and is done much like in the idea behind *attention* [46] as opposed to a sequential or "in one go" approach.

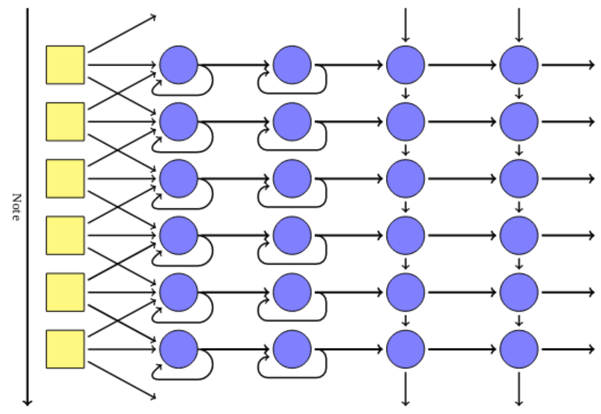


Figure 4.5: Visual explanation of Johnson's biaxial RNN. Lewandowski's RBM visual interpretation is omitted, because to understand the picture, we would need to define used variables and explain it more into depth. For an interested reader, apart from standard Lewandowski paper [40], theano tutorial has a very brief 4 equations long explanation [47]

4.4.1 Enhancing GAN

Standard GAN usually generates samples from uniformly distributed noise from $(0, 1)$ interval. In our attempts, the GAN generator takes the outputs of the LSTM generator (called latent music), or a human composed music part and tries to make changes to it, so it would be harder for the discriminator to distinguish between the real and the fake. As the GAN can easily forget about its input, which in our case is very well aware of time, rhythm and melody, there are some constraints for the generator. The generator operates on already generated music, and thus can rather be called an enhancer.

Firstly we show a visualization of the concept, and then a full explanation of every part is made.

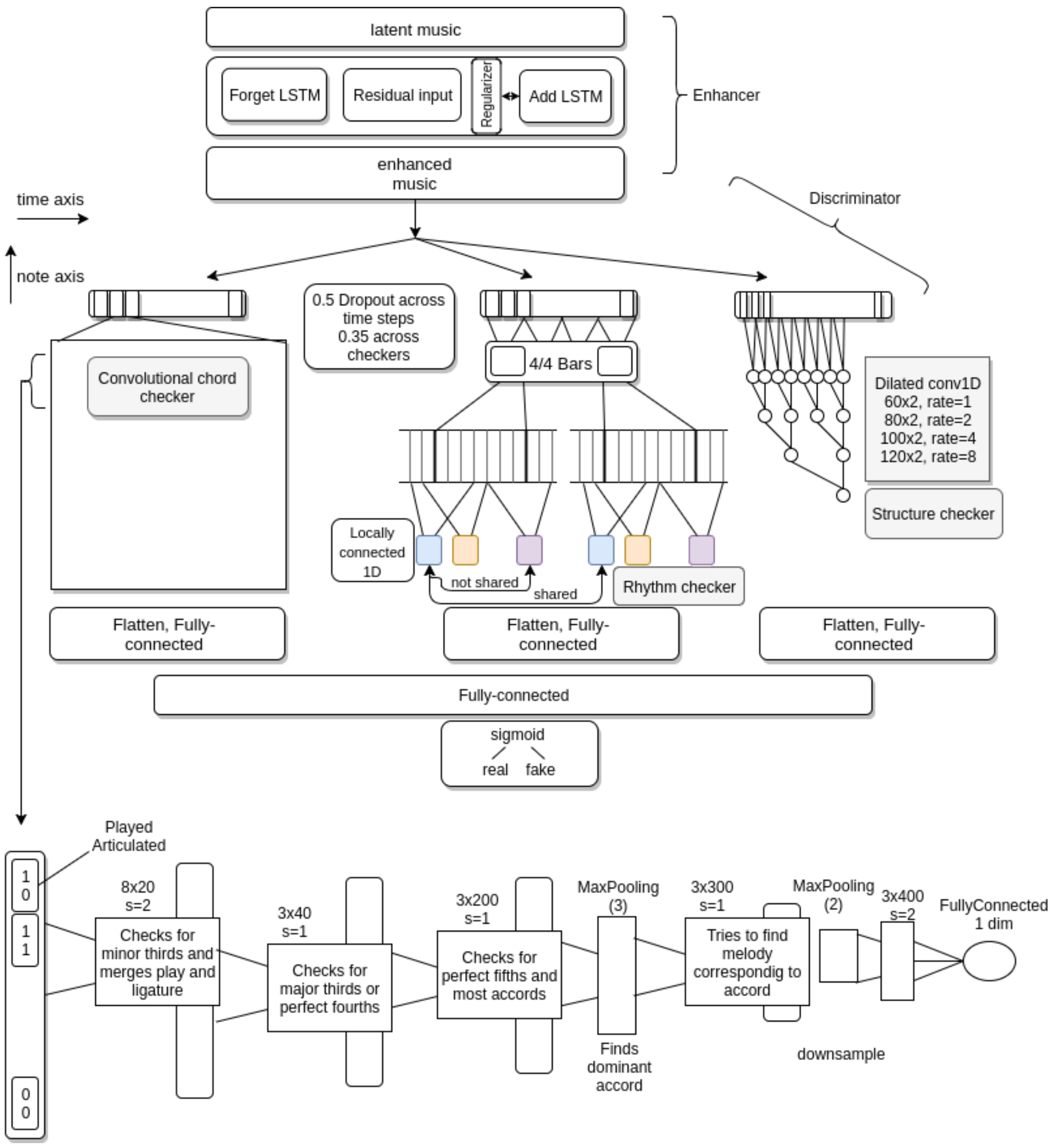


Figure 4.6: The enhancing GAN explained in this work

4.4.2 The enhancer

The enhancer consist of two gating mechanisms. One tries to remove notes that are redundant, out of tune or wrongly put into composition. The other tries to do the opposite, meaning it should add or prolong notes that are missing or that are too short. These gates have regularizers, because of the ability of the enhancer to totally forget about the input latent music and generate one general song that even doesn't sound good, because as we will shortly see, the discriminator which is largely responsible for the outputs does not work with overall ”idea” of the composition.

The enhancer can be regularized with thresholdedReLU, [48] layer which we negate after inference, which is, in case of activation, lowering the effects of the gate to minimum. This our idea has lots of problems with unstable grandients and is yet to be fully usable or rejected, as the results varied.

Another idea that was used and showed better results was to choose a maximum of the outputed probabilities for the changes and let there be a small window e.g. 0.1 for other probabilities to be chosen. This choice was not part of the computational graph as hard assigns are not differentiable, so unlike in usual implementations, discriminator is trained on the generated parts, not the direct sigmoid output of the enhancer/generator.

For the enhancer, we used LSTM, Time distributed Dense or Locally connected layers. Locally connected layers seemed to work the best as they can be trained for every time step independently and not generalize them.

4.4.3 The discriminator

The generating LSTMs for the latent music are doing a good job overall with a very few flaws so adding a recurrent neural network to the discriminator could cause it to remember the main ideas of songs and label them as real/fake according to that. To make the discriminator focus on technical, out of tune melody or harmony flaws, we need to cover the time axis in carefully long windows to trade off the flaws versus a complete musical concept, that we don't want to break. We chose window sizes of maximal length of one musical bar/measure. We feel that it is a balanced length to compete the ”unrealistic” passages but not to break the good coherence of LSTM generated latent music through remembering exact songs. Known regularizations, such as dropout [49] or L1, L2 penalties [50], are also important and were used for every time step in the song and also across all parallel subdiscriminators. We also used a known and simple gradient clipping for both parts of the model to increase stability.

Subdiscriminators

As the discriminator needs to pass good gradients, a wider model instead of deeper was chosen. This can also help the discriminator focus on different parts of music theory independently. The three subdisriminators are explained from left to right as seen in Figure 4.6. They could all be used independently but some can be easily fooled by G if not used together. All of them have a specific meaning in form of subtasks for evaluating/discriminating musical composition. They are used in parallel to fight the vanishing gradients.

The first subdiscriminator, **convolutional chord checker** merges the play and ligature/articulate indices and extracts chord features. It has kernel lengths of the sizes of the most common intervals and chords in classical European music. As every single note is very important, unlike in image classification, pooling layers are placed on the most probable places, where they are means of reducing the computational cost and transposition, but not effectively deleting an important note. Note that this checker can be easily fooled by G through playing correct chords in totally arbitrary rhythm or ”melodies” i.e. pitch continuities.

The rhythm checker first aggregates the time steps into bars and uses a same network on every bar, sharing the weights across them. The network has, for every beat in the bar, a locally connected convolution that checks two steps into history and two steps into future (both history and future can overlay the boundaries of a bar in case of first or last two time steps) and doesn't share weights with other beats. This means that the subdiscriminator is globally learning common patterns visible for individual beats in a measure.

The structure checker features the use of dilated convolutions as in [28]. This should be the main weapon of the discriminator. Stacking kernel sizes of 2 with exponentially growing dilation rates can be thought as deeply checking the current time step and its history up to the frame of one musical bar/measure in a structured way.

WGAN

We use a WGAN, which is explained in Section 2.4, because of the difficulties to successfully train the enhancer (other works trying to achieve similar results showed the same difficulties). WGAN has no sigmoid, nor cross entropy loss function and has a very stable gradient over every stage of training. For the WGAN theory to work, discriminator weights must be clamped around $(-0.01, 0.01)$ values, so the need to use dropout [49] or $L1$, $L2$ regularizations [50] gets lesser or obsolete as they are technically trying to keep the weights low.

For the enhancer, it is needed to use gradient clipping [16] with clip value ≤ 1 as using higher one can make the enhancer untrainable.

Another interesting idea was the use of convolutions not only as the feature extractors, but also to make them a part of final output decision with a direct connections to the output node.

4.4.4 Future work with GANs

Using GAN architectures, and especially our new idea of enhancing GANs, for music composition is a very open field and the works we cited weren't producing very solid results[4], we didn't find a decent success with them. However, we think it is a promising direction to work on in future.

[4]this is only author's own subjective impression

5. Data

To successfully train the neural models, we need a good amount of music compositions, for the supervised learning task. Music data formats are mainly distinguished into raw audio waveforms or exact pitches formats. The models work with a standard format called *MIDI* which specifies the pitches played, their durations and velocity (dynamics). Finding or creating the right dataset is a problem of its own as there are many music styles, genres, orchestra sizes, instruments or rhythms and merging all these together would yield a problem that is much harder than the one we are trying to address. Given the related work working with discrete pitch representations, there were attempts to generate classical, jazz, folk or even a specific author samples. We work with pieces intended only for the piano from Classical, Baroque and Romantic era because these feature a good balance of following the music theory rules versus putting an occasional surprising tones and variations into compositions seen in 20th century.

5.0.1 Music dataset options

The most known datasets for music processing in discrete pitch representations are Piano-midi.de [51], Nottingham [52] or Musedata [53]. We used our own dataset and show the reasons for doing so in this section.

Piano-midi.de has a collection of classical music pieces from 18th to 19th century Europe. It is free to download if you attribute the copyright holder. These compositions mostly follow the Major and Minor scale system which we, in Europe are accustomed to and would like the neural network to learn. However, the size of this dataset is about 249 pieces and training on these data makes the model less diverse and more vulnerable to overfitting. It is used in [40] or [12].

Nottingham is a collection of about 1000 folk music last updated in 2003, with well defined train, valid and testing parts. Parts from this dataset are easy folk songs, christmas song, etudes or just few chords played in a rhythm. Performance of the related work [40] on this dataset is usually the best, yet generating the most simple and less attractive tunes.

MuseData contains classical pieces like piano-midi has size of over 2500 compositions. It is free to download but it is needed to contact the CCARH [54] before doing so. However, the amount of data in *midi* format that we use is only about 200, where the rest of the dataset is in the main format used - *musedata* (`.md`) which is not intended to be used for conversions to other formats due to access restrictions. The format that is used for conversions is *humdrum* (`.kern`) but when using official *humdrum* conversion programs or a well known python library *music21*, problems with headers or ill defined rhythms impairing the conversion appear for most of the data. Note only midi-parser subprogram was implemented due to time restrictions, thus the need for conversions.

imslp.org is a very large database of pdf scores with approximately 400 000 scores. All the data is in the public domain so sharing the dataset is not a problem. Many of these scores have a *midi* file attached. The problem is mainly with the scale and naming conventions of the files because the most reasonable way to download such dataset is to list all *midi* files on the website as opposed to *crawl*-ing through the huge scale website of over 7 million pages. Because the midi attachments don't use a naming convention, the files are often named just by the name of the composition and not the author, which is the best filtering option for our need of 18th to 19th century classical piano music.

Yamaha e-Piano Competition dataset [55] contains MIDI captures of $\sim$1400 performances by skilled pianists. We mention it as the best possible choice for a reader wondering what dataset to use for a similar task we do within this work. However, this dataset is was not very known and we found mentions about it only few days before finishing this thesis.

5.0.2 The dataset

The dataset that was used for the experiments and evaluation was downloaded from *free-scores.com*. *Free-scores.com* is offering every piece for free, but in the case of copyrighted data, not for the redistribution purposes. Because of that, to obtain the dataset, the reader has to download it either by contacting the author for an automated script that is available upon request, or manually filtering (in the website's advanced filtering) properties explained ahead. The data is in **midi** format, intended for **piano**, from the Romantic, Classical and Baroque era. The dataset contains about 900 pieces and note that because of a good naming conventions, the data that falls under two or more genres/eras are not downloaded multiple times so there is no need for testing the hashes for duplicities. As the contents of the website might change over time, we include a list[1] of the exact midi files that are part of the dataset. The download script is intended exclusively for research purposes as some of the files are under copyright.

As this part is the most difficult for the purposes of reproducing the experiments, contact the author in case of unclarity.

Another dataset that was also used for the experiments and the automated evaluation is the *Nottingham* dataset discussed in the previous section. This dataset is attached in the accompanying program folder.

5.0.3 Augmenting the dataset

Augmenting the data can provide more information to the model or simplify the problem by reducing the amount of noise through feature extraction.

We used transposition as a dataset enlarger, because the transposed data carry the same information that we are trying to grasp, but with a different absolute pitch positions. The transposed music parts were randomly sampled from transpositions of $+1, ..., +7$ and $-1, ..., -7$. The range for the transpositions was chosen to include all the keys from a certain scale. This enlarged dataset was used only for experimenting with our enhancing GAN model and not the LSTM models, because of time and technical difficulties[2] and the deep learning library choices made in a large timespan for those two implementations. Therefore, the LSTM models which we evaluated by qualitative surveys have much space for improvements in this matter, because the transposed dataset can help very significantly i.e. Liu et al. [56], p. 6, shows 20% accuracy improvements for both training and test sets for predicting missing voices in Bach's chorales.

[1]attached in `music_network/music/reproducibility_dataset_list_downloaded_1_6_2017.txt`

[2]data does not fit into RAM, Keras library has a method fit_generator for coping with it

Another reasonable option we thought of, is the normalization of keys across all octaves i.e. finding the key that the part is composed in and transposing the part to match a default key within a fixed octave by a difference of these two keys and octave numbers. This would reduce the problem by the scale of all pitches on the piano where the variance of absolute note positions in generated music keys would be ensured by random sampling. This augmentation was not yet tested and is left for the future work. One possible problem is the functionality of the nowadays programs for determining the key of compositions and that many compositions change their keys in their course.

6. Evaluating generated music

In this chapter we are going to see and discuss problems and possible approaches to comparing the results of generative algorithms for music. We are also going to present our own scheme, which was used to get the qualitative and quantitative results.
While computer-generated art is gaining attention in recent years, it is still not very clear how to compare the performances of the models in the area of arts. The resulting music, picture, literature etc. should be unique, so evaluating on validation and test sets does not capture what we want the model to do, and furthermore, the quality of a work of art is always a subjective thing. If we leave aside the paradox of evaluating creativity through rigid metrics, it is rather hard problem to find the right metric (the most commonly [40] [57] used is log likelihood) and it should be on more focus.

6.1 Subjective impression

Subjective evaluation or "ranking" of the author, or a reviewer is a qualitative kind of evaluation. It is not necessarily a bad evaluation, but can be hardly taken seriously, because of the high subjectivity. On the other hand, because of the amount of time the authors spend with their works, they can have a different perspective or be criticizing towards their work and point out the hidden flaws that are important. Therefore, it should be, according to author's opinion, at least a part of the evaluation procedure.

6.2 Listener Surveys

The use of surveys is a good way to average over the subjective opinions to converge on a more robust overall quantitative result. However, while evaluating the models can have rather telling results, a comparison between different works/authors without a centralized authority can be challenging. The surveys also must be carefully designed with the right settings (i.e. form of questions, ordering of listenings, composition of listenings, shaping expectations, number of listenings, length of listenings, knowing that the questions are about computer generated music, number of respondents etc.), which is crucial and can heavily influence the results. Surveys can help us answer main question such as:

- Can people distinguish between real and generated samples?
- Do people enjoy the generated samples overall?
- What characteristic does generated music lack? Where does it shine?

6.3 Automated metrics

Automatic evaluation of generated sequences with discrete outputs is often done using cross-entropy (or equivalently perplexity), e.g. in language modeling. The loss/cost is clearly defined and comparable, but for evaluating music generation, it has certain logical flaws. In music generation, given a time step that we have to generate, there is a large number of options that are approximately equally good. That's not a problem in a training phase (e.g. our experiments in this work or e.g. *softmax* loss for distribution modelling, Policy-gradient (REINFORCE), Sutton [58]), but in evaluation, where we have a problem with interpretation. There can even be an option, that was seen the least in training data (e.g. a disharmonic jump from a monotonic session of the composition) and it could be the very best option from artistic perspective. Rigorously evaluating the performance with multidimensional binary cross-entropy against another work of art from the testing dataset is effectively punishing the model for creativity. From different perspective, that is, because the more acceptable options we have, the lesser the probability for the actual resulting state is, thus producing bigger losses while compared to a fully binary golden validation composition.

6.4 Evaluation procedure

In this section, in light of the previous discussion on the merits of various evaluation methods, we present the exact evaluation scheme that was used in these experiments and the reasoning behind them.

6.4.1 The chosen high level scheme

The main distinction is whether to use an automated process with a rigorous well defined metric that can, rather fairly, compare works of different sources, or to use a survey that is commonly used in humanities. The problem with automatic evaluation is with a correct metric for "goodness" of music. In our experiments with evaluating the proposed models on the Nottingham dataset, we used the cross-entropy loss on the test set, evaluating the next time step based on golden history. However, it is unclear how to interpret values of cross-entropy.

Another perspective is to imagine the common view on perplexity, which is the exponentiation of the cross-entropy, as describing the generation of a next time step is like having a x-sided dice roll for us, where x is the perplexity. It is clear that high perplexity is a bad indicator and that most naive or random models would yield it. However, using this metric on models which produce results any reasonable reviewer can mark as exceeds what is necessary. The bigger problem is when the evaluation yields numbers differing by numbers less than 0.01 and yet the results are very different, for reference, listen to the attached[1] examples skip-dense lstm and vanilla lstm. However, it might also be a bad thing to have the perplexity too low as the works of art are usually not very constrained.

We chose to evaluate our results via multiple surveys. The amount of work needed for this method is higher and the number of reviewers or number of generated samples are always targets of speculation. Note that unless having these numbers very large there is a room for subjectivity. However, a subjective opinion on a work of art will always be the one most correct for each person.

[1] `bad_examples` and `music_network/all_survey_data`

6.4.2 Evaluation layout

We used both evaluation by surveys and automated evaluation. For a correct interpretation of the surveys it is very important to clearly describe their settings.

Automated

We evaluated models through binary cross-entropy/perplexity loss averaged by all pitches and all time steps. The dataset used for this evaluation was different from the dataset for the survey evaluation, which is needed to say because the compexity of these datasets differ. The used dataset was Nottingham 5.0.1, which is well defined in terms of *train, valid and test* parts. Eight variants of our models were evaluated on test set.

We evaluated all the variants described in Section 4.3: *baseline-lstm, skip-dense lstm, skip-temperature lstm, skip-longrange lstm*, all with 0 and 0.2 dropout.

In the equation for computing the binary cross-entropy loss,

$$-\frac{1}{N}\sum_{n=1}^{N}[y_n \log \hat{y}_n + (1-y_n)\log(1-\hat{y}_n)],$$

$$N = T \cdot P \tag{6.1}$$

where T is number of time steps, i.e. 400, and P is our arbitrary chosen pitch span 78 multiplied by 2 for ligatures/articulations, i.e. 156. In computation of the loss we penalized the wrong ligatures/articulations, however, we didn't penalize the network for adding ligatures/articulations where the actual pitch wasn't even played.

The exact method for our computation of the test set loss is as follows: for every iteration of the network (i.e. weight update) we compute the loss on a validation sample/batch. We keep a dictionary of the five best validation losses bound to loss for a whole test set, meaning we only recompute the test set losses (costs computational power) if we hit a good loss on validation sample/batch. The average of the five test losses is the reported value in our results. This way, we might not report the best *test* set loss possible, but we do a tradeoff between not knowing the test set beforehand for hyperparameter search and reducing a random nature[2] of correlation between *test* and *valid* losses.

Surveys

Out of four explored models we did experiments with, two best according to the subjective opinion of the author were evaluated by surveys. They were the baseline[3] LSTM model and LSTM with periodically weighted time skip-connections with $L = 48$. The other model, we called "enhancing GAN" was not evaluated in any way, because our experiments were not satisfactory.

We will see that we used four different sources of our survey evaluated experiments. Two *real* "sources" (historical eras), and two generated "sources" from a baseline model and from a described model. Throughout this thesis, we call

[2]comes from author's experience.

[3]With our improvements, such as majority voting and arbitrary transforming the randomness of the ligatures

the examples from the baseline LSTM model as *vanilla*, the examples from the LSTM with periodically weighted skip connections with $L = 48$ as *longrange*, the examples from the training set as *classical* and the examples from the beginning of the 20th century, the avantgarde era as *modern*.

All the examples[4] were normalized using static tempo, dynamics and instrument. For each listening sample, which was 30 seconds long, we asked these obligatory questions in Czech language with a 5 degree Likert-type scale with an additional option saying *"I don't understand the question"*:

- Is the composition...
  - computer generated? (The *Turing test* question)
  - consistent?
  - euphonic?
  - boring?
  - variable?
- Does the composition contain mistakes according to your judgement?
- Is the composition holding its rhythm?
- Doest the composition have a melody?
- Do you enjoy the composition overally?
- Additional free text space for subjective opinion - will be used, additionally to authors opinions, as a qualitative evaluation.

There were **four** distinct surveys, each with different meaning, content and questions.

- 10 examples of only computer generated samples with 5 samples from LSTM model and 5 samples from skip long range connected model, with the Turing test question. We will call it *only-gen*.
- 10 examples of only real examples with 5 samples from the training data and 5 samples from modern avant-garde authors, with the Turing test question. Called *only-real*.
- 15 examples of mixed samples of the two models and real examples, with the Turing test question. We will cal it *mixed-turing*. The composition of these examples was 4 *longrange* (picked the bests from 15 random samples), 4 *vanilla* (picked the bests from 15 random samples), 4 *modern* and 3*classical*.
- 15 same examples as in *mixed-turing*, however, without question about the origin of the composition and the respondent's knowledge of true intentions of the survey i.e. it has a "misleading" introductory text. Called *mixed-calibrate*.

Respondents were asked questions regarding their **age, experience with musical instruments, frequency of listening to music** and **technical and machine learning knowledge**. Using these information, we will see in 8.2.3, how much diverse and in what field we would need to get the respondents in future works.

[4]To reproduce the generated samples follow the section **Reproducing results** chapter in the user documentation attached. The concrete samples in each survey with their exact ordering can be seen in attachment files `music_network/music_poll_table.txt`[5] and folder `music_network/all_survey_data`.

6.4.3 The ordering

The ordering is very important because the comparing factor has a major role in the real vs. generated decision making. There is a bias that come from the fact that in the beginning, the respondents answer mostly by their common sense, but in the end, they may have found some patterns and improve their accuracy. Randomizing the ordering for every one respondent is good to counter the rather biased starts and endings of the survey.

On the other hand, the static ordering has an advantage that all the respondents get exactly the same conditions. That is particularly important if the number of respondents is rather low, because we would need a very large amount of respondents to see how much a sample can "confuse" the listener in terms of the *Turing test* question for every position in the ordering. That amount of respondents is not something we have. The bias problem could be fought by calibration listenings that illustrate the range of met samples and prepare the respondent, and which the other samples can be compared to. We added two of those illustrative listenings at the beginning of the *mixed-turing* survey.

We decided for both the samples and the related questions in our surveys to be ordered statically. Although the respondents could go back to previous samples and change their answers[6], due to the length of the survey, they probably did not.

6.4.4 Reasoning behind the evaluation scheme we used

The reasons for using surveys for music evaluation was discussed clearly in this chapter, however, the form which we are proposing needs some explanation. The reasons for mixing the real and the generated compositions into surveys and trying to find differences in various characteristics are very straight forward, but what about the expectations or preconceptions? The proposed scheme is a part of the experiment, just like the explored models, and we want this experiment to capture as much data as possible. Therefore, considering the influence of the preconceptions about A.I. on the ratings, we want to show exactly how much they affect the responses in the survey, so we feature two versions of surveys with the same examples, same ratings of the characteristics, but without the *Turing test* question and a "misleading" introduction text.

The next two additional variants of the surveys, *only-gen*,*only-real*, are important as well. The expectations of the listener on the proportion of real/generated examples could bias the ratings and not only for the first few samples. We need to know how strong this effect actually is, or if it really exists. We need to find boundaries shows us of how many samples is the listener willing to mark as generated/real, even if all the samples are actually generated or all real. Because we created all the surveys in the same time, we predicted that the respondents will be expecting half to half distribution, for the two main surveys.

[6]That would, in the ideal come out, effectively suppress the skewness of the static ordering

7. Experiments

In this chapter, we are going to explain the details of our experiments. As we already did some explanation on our internal experimentation in previous chapter, here, we only explain details on the experiments directly leading to some form of rigorous evaluation. Accompanying program and its documentation are attached attached.

7.1 Dropout

Dropout is a regularization method that uses an idea of ensembling random sub-models to counter overfitting [59]. During each training step, a percentage p of neurons (e.g. 50%) is being zeroed out (practically removed from the network). This way, the network is immune to fragile dependencies found in the training data and is better at generalizing. During inference, all weights must be multiplied by a complement of p, to prevent overexcitement of neurons, that are used to lower the weightsums from training phase.

7.1.1 Dropout in generative RNNs

Dropout is a strong and popular method for neural networks which try to capture a general concept. However, we have doubts about its use when it comes to generating music. It is wanted, from the model, to capture the most common chords and their transitions and the kind of generalization we would like, which is their absolute pitch translation invariance, is not evident from the way dropout works. It is clear that when we hear a generated sample which is just a remembered training sample, we would have to deal with it, and LSTMs are known to be used with dropout for their overfitting capabilities. However, we will see in Chapter 8, that the samples were lacking coherence and consistency, which is not a sign of overfitting the real data which has a musical idea encoded within the notes. Unlike in e.g. image classification with colored pictures, we only have a binary ”2d picture”. Where the need for generalizing in the huge color space times the size into one class is high, we don’t assign any classes and the more possibilities for interestingly developing a musical idea, the better. There is not that much need for the generalization in sense translated from common use of dropout. The most needed generalization we would like to get is the note axis invariance, as the chords are, unlike e.g. picture of a penguin, rigidly defined. This kind of generalization could be, in a future work, possibly accessed by a new representation of music for machine learning.

7.1.2 Overfitting and our dropout scheme

We should also state that we used a training scheme [44] which introduces noise into gradients, effectively making the model less prone to overfitting [60]. With the use of usual mini-batch training, we found few testing samples resembling individual training data (i.e. not remembered exact notes, but stayed with a ”musical idea” already seen). With the online-training with ”gradients” noise, according to comments of our respondents and our opinions, we didn't have noticeable problems with overfitting.

The most commonly seen dropout values in similar models are 0.0 [40], 0.2 [61] [35] and 0.5 [12]. For us, dropout significantly increases the training time, which is intuitive provided how this technique works. We evaluated our models 8.6 with both 0 and 0.2 dropout versions and found minimal differences. We also felt that higher dropouts resulted in under-learning by the network.

7.2 LSTM training

In our experiments, we trained these models:

- A baseline LSTM model, later called textitvanilla
- LSTM with dense skip-connections of length 16 for every time step, called *skip-dense lstm*
- LSTM with skip-connection of length 16 for every time step, but with periodically weighted connections adding more information flow to the beginning of musical bars, called *skip-temperature lstm*
- LSTM with longrange skip-connections connecting with length 48 connecting residual gaps across 3 musical bars and also featuring periodically weighted skip connections, called *longrange lstm*

All of these models were trained along with a second neural network for regression of the amount of polyphony that we described in previous Chapter. The second network didn't use skip-connections.

From our training dataset, we used only music played in 4/4 beat because of our music representation and rigid architecture of our model. Therefore, our amount of data used for training was lowered. Interesting insight is that using data played in other beat didn't produce much worse results, however, stuck to the correct beat for our data. We used first 400 time steps i.e. 400 durations of 16th notes. With no padding, but cutting longer and skipping shorter training sequences.

We described most of our settings i.e. hyperparameters in previous Chapter about models, however, we are going to add few more infomation about our experiments. We used online training [44], because it was faster trained and has benefits when the amount of training data is low. We used random sampling of training data, because we thought that for our problem it could bring more interesting results.

For all our networks trained on Nottingham dataset, we used 10 thousand iterations (weight updates). And for the two models, we evaluated through surveys (*vanilla* and *longrange*), which were trained on *free-scores.com* dataset, we used 15 thousand iterations. We used *adadelta* [62] optimizer, which one of the advanced optimizers with minimal overhead over SGD and requires no manual tuning of learning rates. We didn't use any gradient clipping, because our model was not having troubles with training.

8. Results

In this chapter, we are going to look at the results. The results from authors perspective, automated evaluation results, and results from the surveys discussed in evaluation Chapter 6.4.2.

For survey evaluated experiments, we had data from 4 different sources, two *real* "sources" (styles, historical eras), and two generated "sources" from a baseline model and from a described model. Throughout this thesis, we call the examples from the baseline LSTM (mostly baseline[1]) model as *vanilla*, the examples from the LSTM with periodically weighted skip connections with $L = 48$ as *longrange*, the examples from the training set as *classical* and the examples from the beginning of the 20th century, the avantgarde era as *modern*.

We released four different surveys to rank our resulting samples. The surveys were in the Czech language. All answers were obligatory so we have roughly the same amount of votes for every sample. The samples and the questions were statically ordered, so there was a problem with the discussed **??** biased samples, due to the comparison factor. Two of the four surveys are for calibration of the possible respondents expectation skewness, which we call *only real* and *only gen*. One that has 15 evenly distributed examples from the four classes of origin, which we call *mixed-turing*. The last one with exact same examples as the previous but with no question about the generative origin of the example (sometimes called a *Turing test* question or a *real-likeness rating*) and a "misleading" introduction text. This one was proposed to verify the rankings from the *mixed-turing* survey on various characteristics and the choice of the surveyed characteristics. We call this survey *mixed-calibrate*. The amounts of complete responses were 12, 11, 32, 33 respectively with minor to no overlapping of respondents across surveys.

We provide the answers from the surveys to the main questions in the form of standard boxplot and violinplot graphs with the means added as red points showing the main differences.[2]. These graphs are commented in a way that allows the reader to very quickly interpret them. It is needed to say that the comments can be of miscellaneous nature and provide possible reasons why no interpretation is possible or why some interpretation is highly possible.

We add the miscellaneous comments mainly because of that our evaluation procedure of generated music is maybe even bigger experiment than the newly added periodically weighted skip connections to the LSTM. That being said, we add conclusions on the evaluation scheme and how much it helped us answer the main questions:

- Can people distinguish between generated and real samples? (*accuracy*)
- Out of all samples marked as generated, how much of them really are? (*precision*)
- Out of all generated samples, how much of them are revealed? (*recall*)
- Do people enjoy the generated samples overall?
- Do people enjoy the generated samples more than the real ones? Which ones?

[1]With our improvements, such as majority voting and arbitrary transforming the randomness of the ligatures 4.3

[2]The used *csv tables* containing the results are attached, in case the readers finds their own query of interest. `surveys` folder, all of the files with *csv* extension

- What characteristic does generated music lack? Where does it shine?
- How much do people agree in main characteristics?
- How correlated are the chosen characteristics?
  - Which correlate to the overall enjoyment?
  - Which correlate to the impression of sounding generated?
- Is there a difference between the two implemented models *vanilla* and *longrange*?
- Is there a relevant feature of the respondent, such as: plays the musical instrument, listens often, has technological background?
- Was there a difference in ranking or in strictness in surveys *mixed-turing* and *mixed-calibrate*?

8.1 Quantitative look at the results

In this section, we are going to look at the possible answers to the questions listed here 8. Moreover, we are going to try to answer what does the results reveal about the evaluation method alone. The main questions are:

- How much can the rankings be trusted? Is this a good method for music evaluation? Under what conditions?
- Did the calibration reveal anything about the expectations of the respondents?
- How did rankings on the same examples change when:
  - we told the people that some examples might be generated? Are people kinder to computers?
  - the respondents marked a composition as generated?

8.1.1 Calibration results for the main survey

We used two of the four surveys for verifying our chosen ratio of real and generated samples into the main survey. The goal is to find the boundaries of the willingness to mark certain portion of samples as real or as generated. This helps us to interpret the main results more correctly.

In the main *mixed* surveys, we distributed the real and generated samples evenly with an expectation that the respondents would expect just that, thus making the effect of this "prior on generatedness" as small as possible. We found that in both *only real* and *only gen* versions, the first half of the responses were rather correct in identifying the origin of the samples, but in the middle of the survey, the influence of respondent expectation won over their senses as seen in the results for both calibration surveys in 8.1. According to the calibration results, we hypothesize that the chosen balance of the real/generated examples was reasonable for our samples.

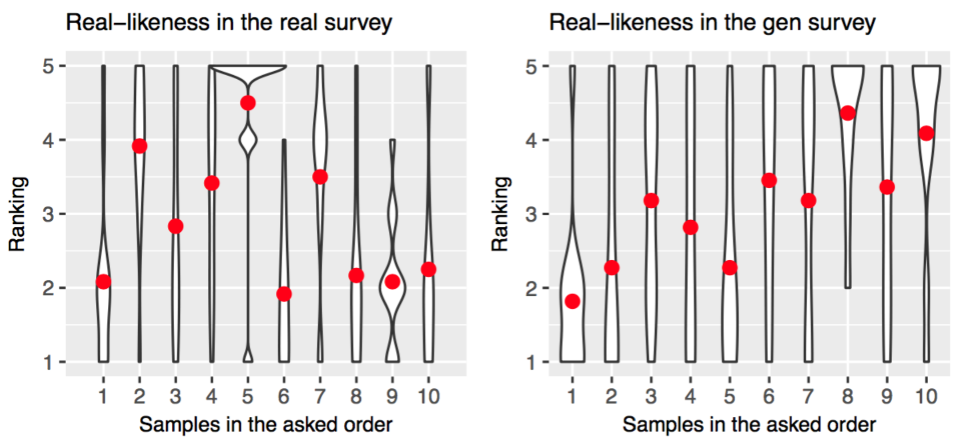


Figure 8.1: Tendency to expect evenly distributed samples. First few samples might appear biased because we didn't add the preview samples illustrating the range of possibly met samples, as unlike in the *mixed* survey. This means the respondents could not compare the first samples to anything. As this is the first *violin* plot we see here, the explanation is: the red dot stands for the mean, the shape illustrates the distribution of votes, and in all the possible characteristics, the higher value on y-axis means the better result

We also found that in all the characteristics, the respondents were getting more strict in the course of *only real* survey and more benevolent in the course of *only gen* survey as seen in our results with all of the 9 characteristics aggregated in 8.2.

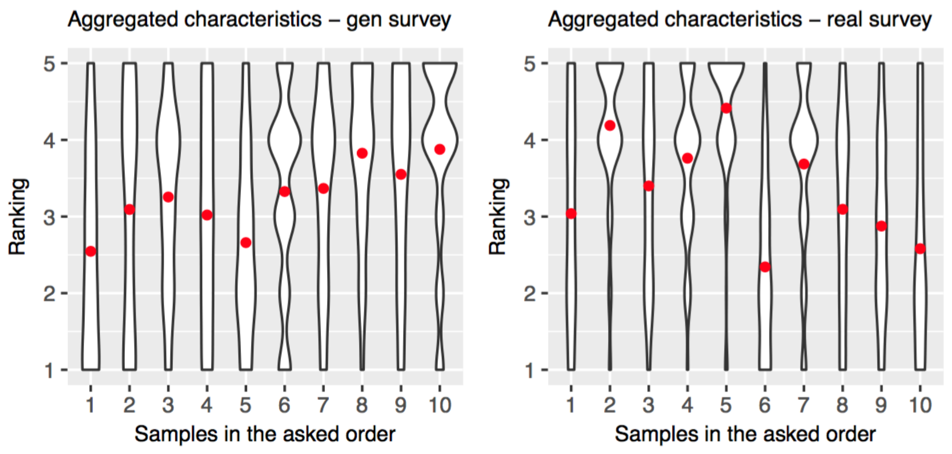


Figure 8.2: The expectation of evenly distributed real vs. generated samples, with all the 9 characteristics aggregated (all followed "*the higher the better*" scale, where "better" in the Turing Test question means the respondent considered the sample real, rather than generated). Results show that hearing real samples forces strictness and hearing generated samples forces being more benevolent.

This only supports our hypothesis, that there really is an expectation about the distribution of real vs. generated samples. The respondents seem to not only expect about a half of the samples to be generated, but also to be worse. That leads to the correlation question between the characteristics and real-like feel of the music, which will be discussed in this chapter Section 8.10.

A possible valuable result from the *only gen* survey is the leaning towards better rankings in real-likeness and overall-impression for *longrange* samples over *vanilla* samples. This result is not that highly influenced by the awaiting factor of the respondents discussed in the previous two paragraphs, because of the evenly distributed samples, though it might be of a random nature, because of the low number of respondents for the calibration surveys. Nevertheless, the most supporting co-result is that the results are the same in both *mixed-turing* and *mixed-calibrate* in terms of the relation *better/worse*[3]. See 8.1.5.

8.1.2 The capability to distinguish between the real and the generated

We found the accuracies for the origin question in the 3 of the 4 surveys, that asked the *turing test* question. The ability to distinguish was lower for the *only real* and *only generated* surveys, which was nothing unexpected because of the expectations of the respondents to see evenly distributed samples. We used both hard-decision and scaled accuracy, where in hard decisions, the responses saying *half-to-half* were omitted and in scaled decisions, the weights were computed such as this: *Yes = 0.9375, More like yes = 0.75, half-to-half = 0.5, More like no = 0.25 and No = 0.0625.*

Type	**Survey**	**Accuracy**
Hard-decision	Only real	43.52%
Hard-decision	Only generated	47.57%
Hard-decision	Mixed	61.69%
Scaled	Only real	46.77%
Scaled	Only generated	48.18%
Scaled	Mixed	57.31%

Table 8.1: Accuracy with various settings

We will later see, that the *modern* real data fooled the respondents better than the *classical* real data and that the *longrange* generated samples seemed slightly more real than the ones from the *vanilla* model. It is clear how the accuracy will move against the joint accuracy when comparing these two pairs: *modern* × *longrange* and *classical* × *vanilla.* How much and in what survey settings can be seen in the following table:

Hard-decision	Only real	81.39%
Hard-decision	Only gen	53.84%
Hard-decision	Mixed	63.63%
Scaled	Only real	70.83%
Scaled	Only gen	54.09%
Scaled	Mixed	59.65%

Table 8.2: Easier to distinguish subsets - *classical* × *vanilla*

Hard-decision	Only real	18.46%
Hard-decision	Only gen	41.17%
Hard-decision	Mixed	59.91%
Scaled	Only real	30.55%
Scaled	Only gen	41.81%
Scaled	Mixed	56.67%

Table 8.3: Harder to distinguish subsets - *modern* × *longrange*

[3]not in terms of exact numbers, because the strictness of the responses varied, because of different survey sample proportions, questions and a purposefully missleading introductory text

As the accuracy cannot tell us what kind of errors were made, we can also observe the precision and especially, the recall of the generated samples. A complement of recall can show us a percentage of respondents who classify the generated samples as real. To better interpret the shown values, we also provide a distribution of the guesses. The scaled and hard-decision versions are computed exactly as in the accuracy results.

Type	**Survey**	**Precision**
Hard-decision	Mixed	63.63%
Scaled	Mixed	59.65%

Table 8.4: Precision for the generated samples, calibration surveys have trivial precision, because the samples are all of the same *real/gen* class

Type	**Survey**	**Origin**	**Recall**
Hard-decision	Only generated	Agg. generated	47.57%
Hard-decision	Mixed	Agg. generated	59.15%
Scaled	Only generated	Agg. generated	48.18%
Scaled	Mixed	Agg. generated	55.30%
Hard-decision	Only generated	*longrange*	41.17%
Hard-decision	Mixed	*longrange*	54.71%
Scaled	Only generated	*longrange*	42.84%
Scaled	Mixed	*longrange*	52.6%

Table 8.5: Recall for the generated samples, illustrated for *longrange* samples as well

Recall is probably the most important indicator for our purposes. We can see that in the survey with half-to-half ratio of real and generated samples, $\sim$ 42%[4] of the responses marked the generated samples as coming from a real source.

When we focus only on the proposed *longrange* model, $\sim$ 47% of the responses marked the *longrange* samples as real, in the main mixed half-to-half survey. These results are a rough average of taking the uncertainty of the respondents into account and not taking it into account, however these results do not substantially differ. For the baseline *vanilla* samples, 39% of the responses marked theam as real.

To better interpret the results and their trustfulness, the overall ratio of *Turing test* guesses for the *mixed-turing* survey were distributed like this: 46% for real[5], 42% generated, 12% half-to-half and 0% undecided. This distribution shows a relatively high percentage of uncertain classifications[6]. The ratio of real/generated guesses was slightly favoring the real guesses, which is a good result for our efforts to fool respondents.

[4]if we average up on both scaled and hard indicators

[5]both certain and uncertain variants merged

[6]to better show the amounts of uncertainty for various classes of sample's origins, we use *violinplots* which works similarly to heatmaps

Overall results with grouping all samples by their origin is seen in the following figure.

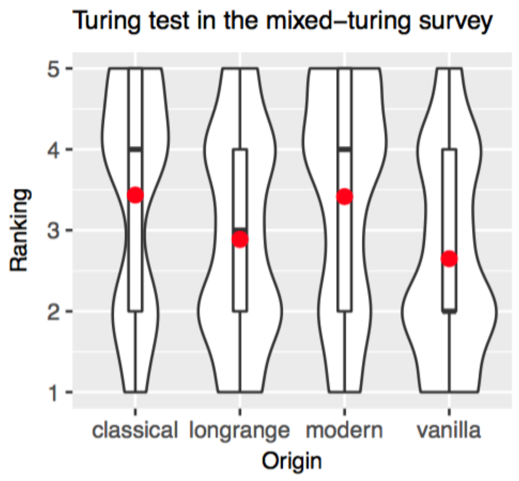


Figure 8.3: The real-likenes or *Turing test* by the origin of the samples

According to the respondents, the *longrange* samples seemed more real than *vanilla* samples. The results also show no clear difference in the real-likeness of the *classical* and *modern* real samples. That is an interesting result, because we added the modern samples to better fool the respondents, because of their unconventionality and frequent atonality.

8.1.3 The fineness of the generated music

In this section we are going look at the two main things. How the generated music stand against real music in overall impression and other characteristics, and how the two evaluated models stand against each other. The next important thing is where the generated music, generally or in *vanilla* and *longrange* models exclusively, shine and where it lacks. We are not only going to pick the best/worst rated characteristic, but also compare it to the results of the other model, or even the real samples results, where we could actually find a completely different best/worst characteristic[7] and verify the ratios in the *mixed-calibrate* survey, to provide the most possible interpretation, or hypothesize that no good interpretation could be presented.

What is the overall impression for the generated examples

In this section, we will merge the four origins into two major ones - generated and real. We will compare it by overall impression and in the next section, by selected characteristics and all characteristics aggregated.

[7]that is, because the best rated characteristic in the model, can be proportionally very bad against real samples and possibly the worst rated characteristic can possibly be on par or better compared to the real samples results, meaning it is a good result.

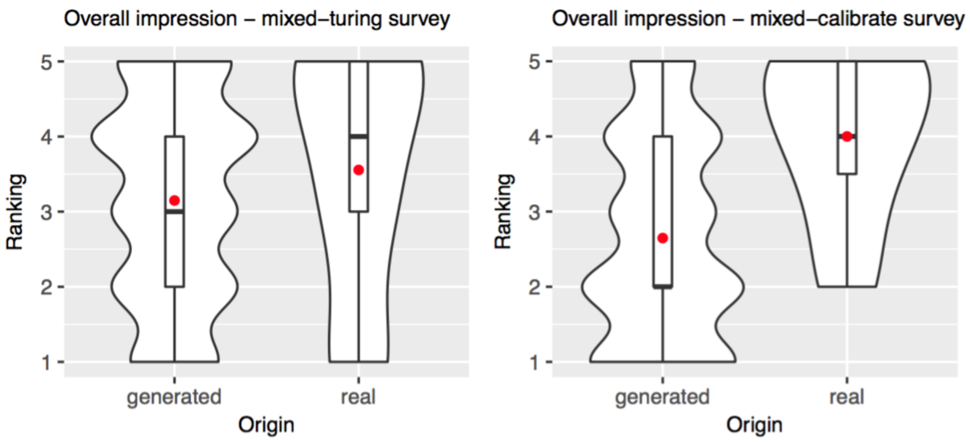


Figure 8.4: Overall impression by the two significant *mixed* surveys. In these graphs, we firstly see a boxplot attached, so an explanation is appropriate. *Boxplots* further help us understand the distribution of votes by dividing the overall votes into 4 sections with the length of $\frac{1}{4}$, where the dividers, i.e. quartiles, illustrated by horizontal lines. It can happen that the sections are merged in one point.[8]

In the Figure 8.4, we can see a expected result: the generated samples are significantly worse at overall impression than the real samples.

Another interesting point is the bigger difference in the *mixed-calibrate* survey. That is an observed tendency in all of the characteristics caused by the missing question about the origin (*Turing test*) and a "misleading title". We observed that the people are *"kinder to computer"*, either because of knowing they might be dealing with generated music, or either if they have already answered the *"Turing test"* question positively and have a completely different perspective, possibly a one where they think: *"for a computer? pretty good"*. We mentioned this as it is just putting a seed in the reader's head at the first sight, however, we will discuss it properly later in 8.2.2.

Comparing the characteristics

The last observation with the examples from the two models merged, and the real examples from the two origins merged, is going to be a comparison of the proposed characteristics.

In Figure 8.5, we can see that the differences between the two surveys (compare column wise) are very low, and apart from being transposed to lower absolute rankings in the *mixed-calibrate* survey, they mostly agree. The exception is *fluency* where the averaged results do not differ in absolute rankings and it is also on par with the fluency of the real music samples in both of the surveys which is a good result.

When comparing the real and the generated samples (compare row wise), we can see one big difference in the variability of the samples. Where in every characteristic, the real samples were ranked slightly better, the real samples seem less variable, but we cannot conclude if this is good or bad for the overall impression. However, even if it seems like a major differing characteristic between the two

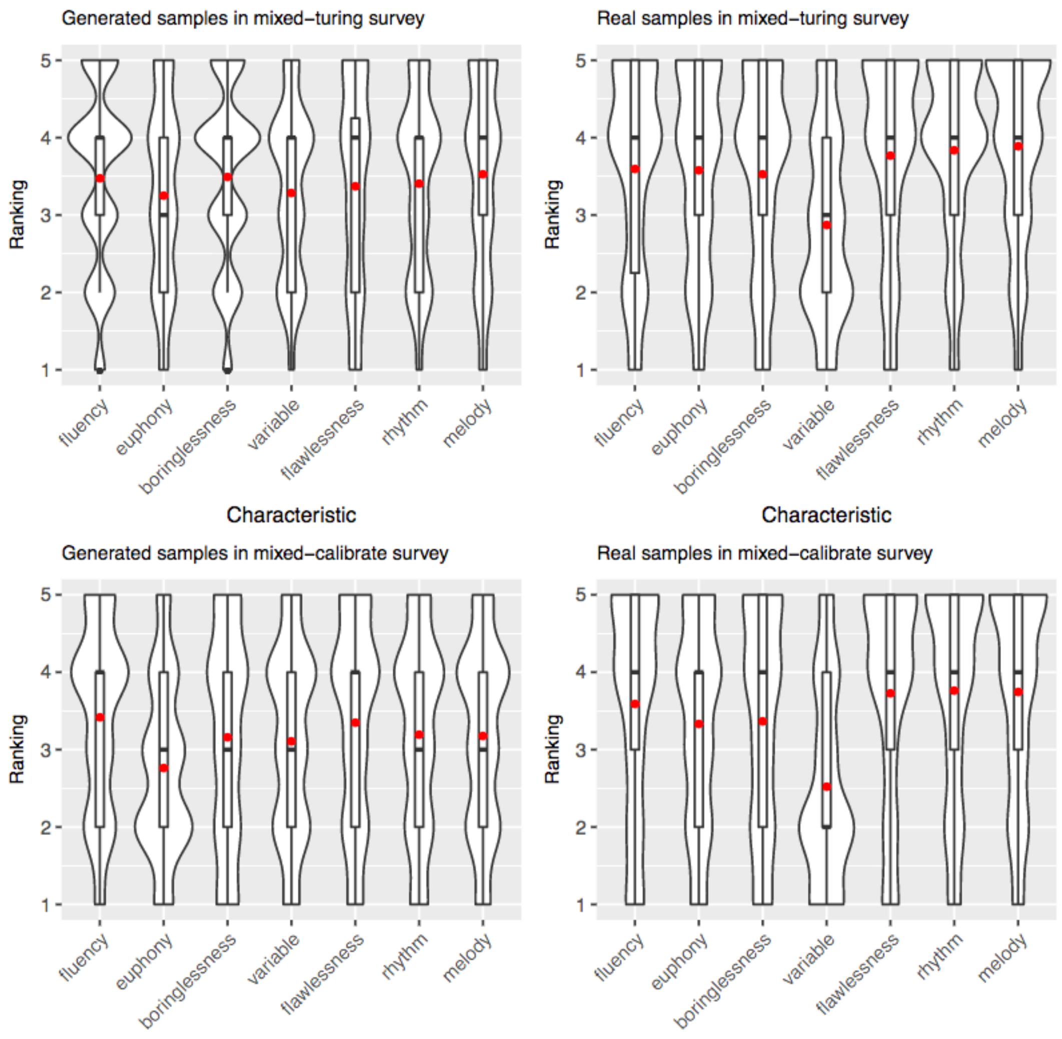


Figure 8.5: Comparison of the real and the generated music through surveyed characteristics, in both mixed surveys. Both the surveys have exactly the same samples and roughly the same number of respondents, as specified in 8. Therefore, this comparison is reasonable.

classes[9], we will see that the correlation between the variability and the *Turing test* question the lowest observed, just like the correlation between the variability and the overall impression, see 8.10. Therefore, we hypothesize that the respondents did not focus much on the *variability* characteristic when thinking about the possible origin or the impression of the sample, even though it could be a major indicator.

8.1.4 The best generated sample

One of the generated samples got into the first third[10] in two most important characteristics i.e. *Turing test* and *overall impression*, outranking a number of

[9]This is also stated by some qualitative comments by respondents 8.3.1

[10]The graphs supporting this claim are attached: `surveys/mixed_real_likeness.pdf`, `surveys/mixed_calibrate.pdf`, sample name: `music_network/all_survey_data/example_longrange_3`

real samples. This result is shared in both *mixed-turing* and *mixed-calibrate* surveys.

An important question behind this result is *why*. How was it different from the other samples from *longrange* and *vanilla* models?

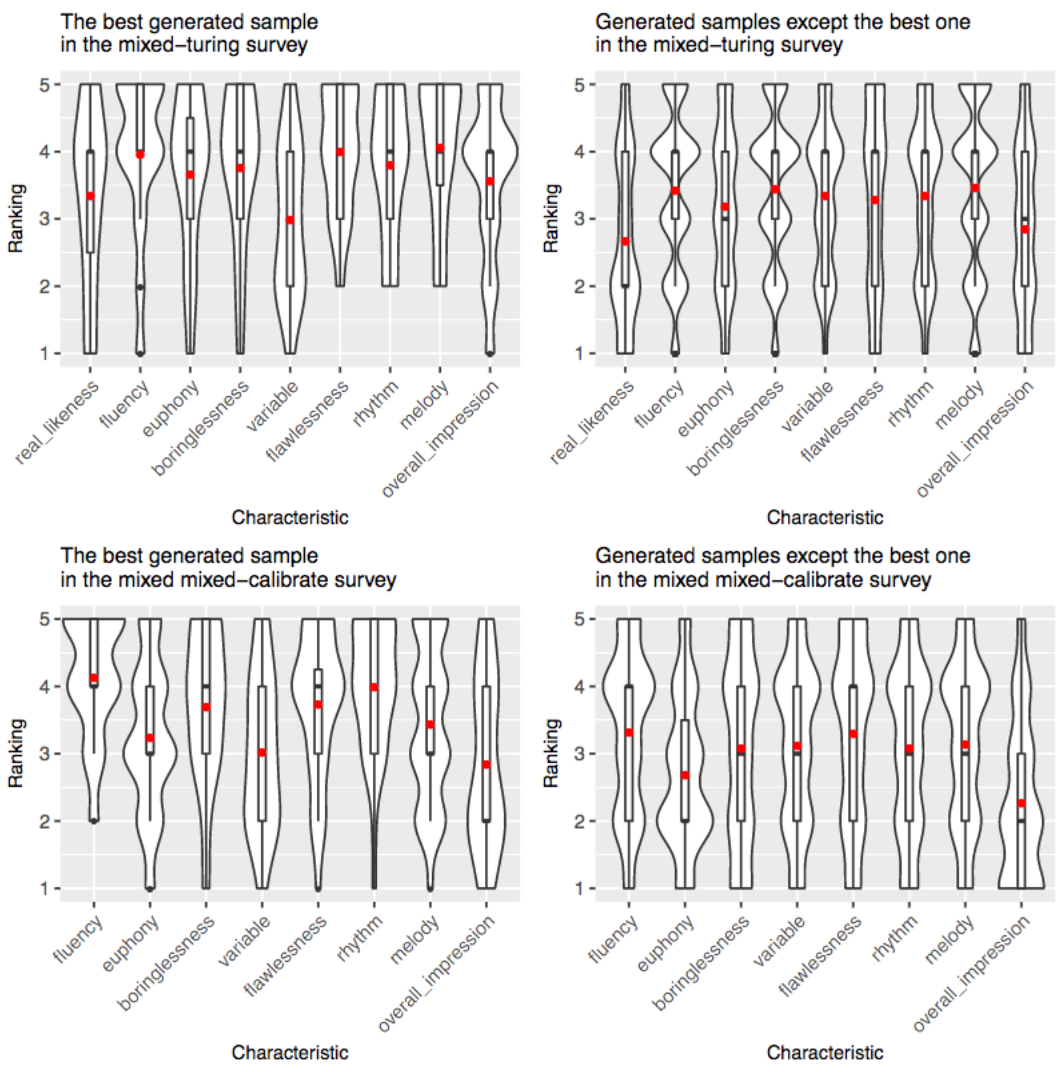


Figure 8.6: A comparison through all the surveyed characteristics, of the best generated sample in terms of *Turing test* and *overall impression* and the rest of the generated samples through surveyed characteristics, in both mixed surveys. Both the surveys have exactly the same samples and roughly the same number of respondents.

In Figure 8.6, we can see that the best sample more or less better in every characteristic, except the *variance*. We will later see that in all our data in both the surveys, the *variance* has the lowest correlation to both *Turing test* and *overall impression* and the other characteristics have a positive correlation to them., see 8.10. The seen results only add to the found correlations.

8.1.5 Differences between the two evaluated models

Firstly, we are going to compare the *Turing tests* and overall impressions and then find the most important characteristics, where the two models shine, lack and differ.

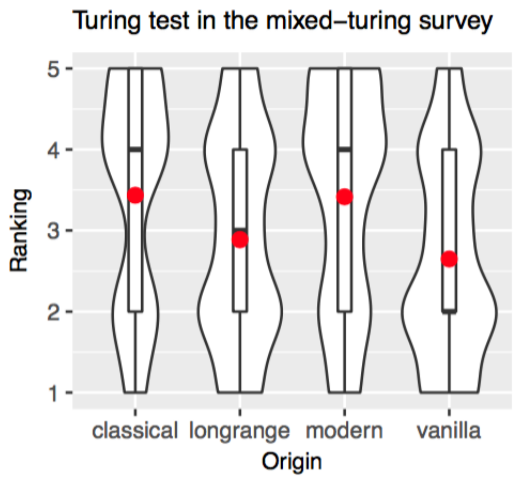


Figure 8.7: The real-likenes or *Turing test* by the origin of the samples

In Figure 8.7, we can see that the generated samples from the *longrange* model were considered slightly more real, than the ones from the baseline *vanilla* model. The second survey, the *mixed-calibrate*, didn't ask this question so we cannot verify this further with a second independent survey with same amount of respondents. However, from the *only-gen* calibrating survey, unfortunately with just a low amount of respondents, we saw the same results, see 8.1. We can also see that the samples from the two real origins seemed similarly real, with a difference that for classical music, the responses answering *half-to-half* were rarer.

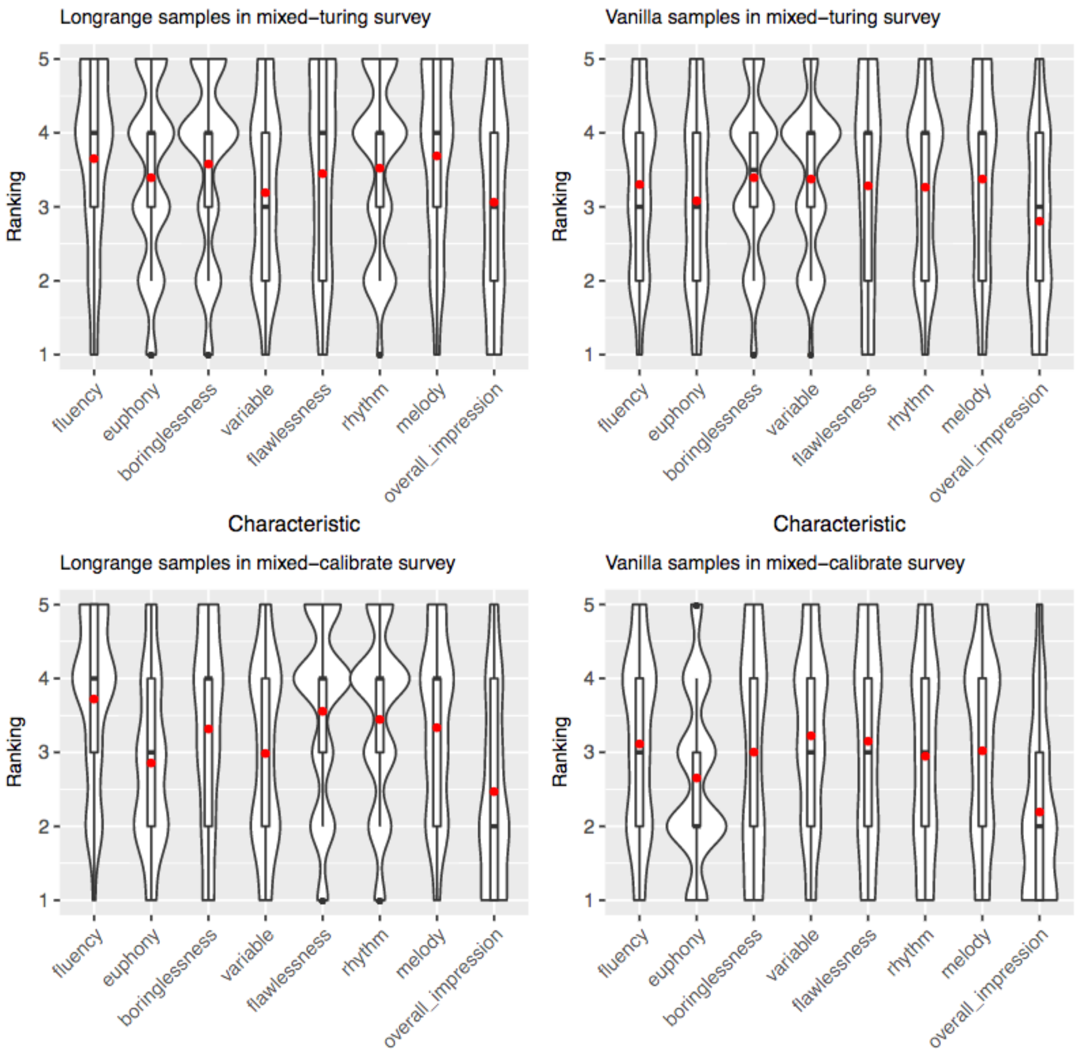


Figure 8.8: Comparison of the *longrange* and the *vanilla* samples through the surveyed characteristics, in both the mixed surveys. Both the surveys have exactly the same samples and roughly the same number of respondents, as specified in 8.

The Figure 8.8 is very important, because it helps us understand where the two implemented models differ and if it is worth it to do a comparison of the ”better” model and the characteristics of real samples.

We can see, that both the surveys proportionally agree (compare column wise) and that the biggest absolute difference is in the *overall impression* question, where the respondents who didn't know about the possibility of generated samples were more strict.

When comparing the samples from the two evaluated models, we can see that according to the respondents, the *longrange* samples were more fluent and euphonic, and less variable, than the ones from *vanilla* model. They were also more rhythmic and melodic. According to the results of the *mixed-calibrate* survey, but not seen in the second survey, the *longrange* samples seemed also more flawless.

Comparing longrange examples to modern music

We've seen that the *longrange* generated samples were closer to the *real* samples in terms of the *Turing test* and the *overall impression* results. Prior to the preparation of the surveys, we felt that the generated samples might be harder distinguish, when compared to the avantgarde music from the beginning of the 20th century, which was not part of the training data. In Figure 8.7, we've seen that there is no real difference between the *classical* and the *modern* samples in the *Turing test* question. Let us examine a comparison of the *longrange* samples to the *modern* ones in other characteristics.

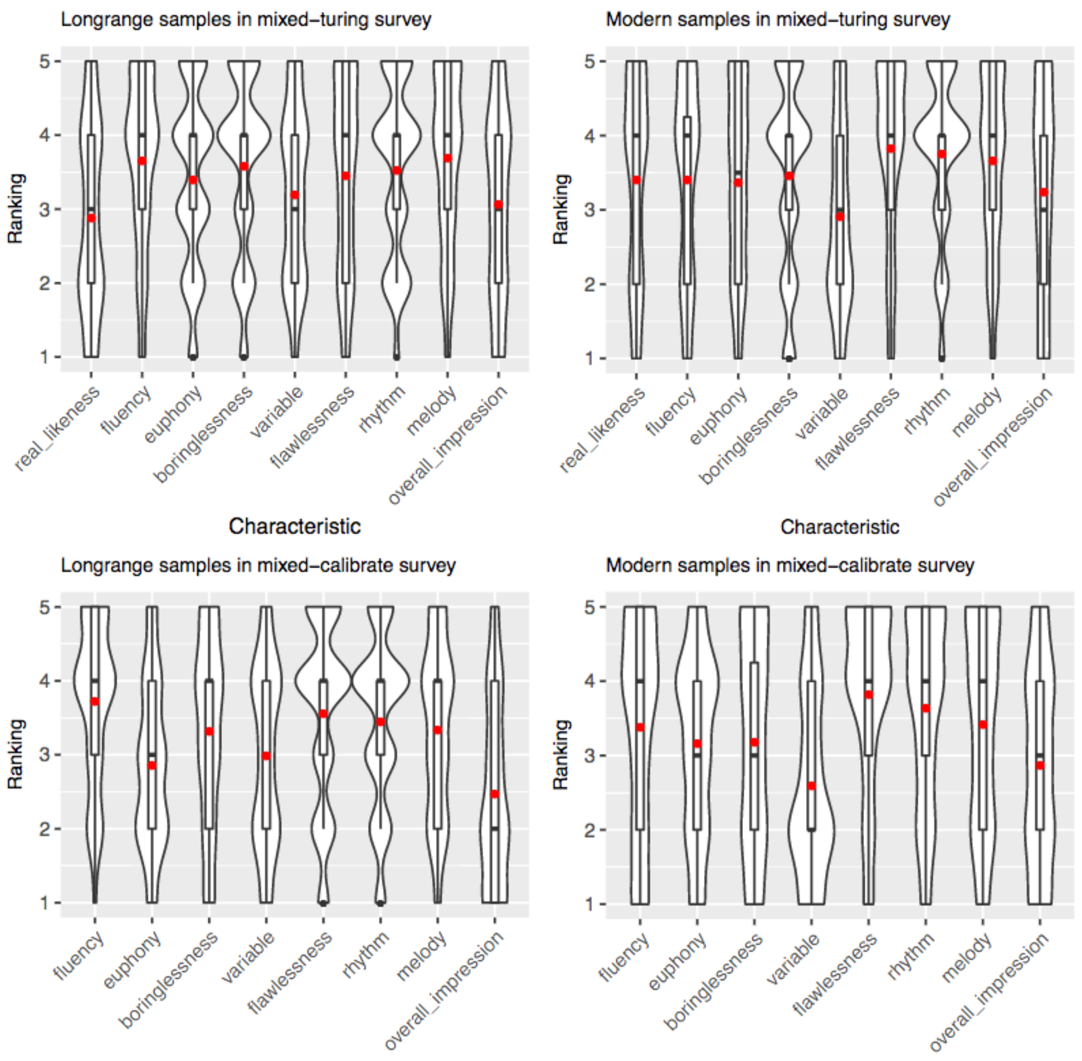


Figure 8.9: Comparison of the *longrange* and the *modern* samples through the surveyed characteristics, in both the mixed surveys. Both the surveys have exactly the same samples and roughly the same number of respondents, as specified in 8.

In Figure 8.9, we see that *modern* music samples were considered more real and more enjoyable. However, there were some characteristics, where *longrange* model outperformed or at least was on par with real samples. These were mettern melody and fluency. The *longrange* samples seemed also less boring.

8.2 Results for the surveys scheme

In this section, we will try to evaluate the proposed *evaluation by surveys*. It is mainly hypothetical, and we discuss the choice of characteristics and trustfulness of the used evaluation scheme.

8.2.1 Correlations of the chosen characteristics

In this section, we are going to see if it is important for a neural network to perform well relatively to some chosen characteristic. It can provide us answers on what we should focus our ideas on new architecture structures of the models. Which characteristic is the most worth it to improve from the *overall impression* and the *Turing test* question perspective? Which characteristic is not that worth it to craft in future work?

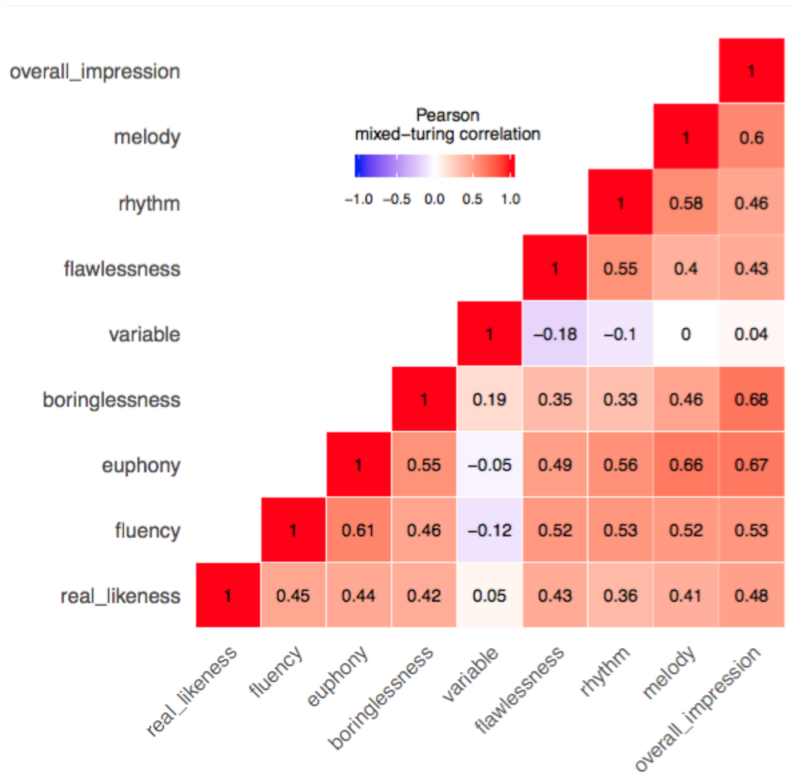


Figure 8.10: Pearson's correlations in the *mixed turing* survey. The *mixed calibrate* survey correlations roughly match the shown and are attached in `surveys` folder.

The most important are the correlations for *overall impression* and *Turing test* question. Correlation coefficients of all the characteristics to *real likeness (Turing test)* except the *variance* shows mediocre positive correlation. For *overall impression*, the three characteristics that are more correlated are *melody, euphony and boringlessness*[11]. Improving these contribute the most to overall impression and in Figure 8.5, we have seen how good the generative models were in them.

[11]As the author, I am convinced that if we didn't add the beat representation into the LSTM models, the surveys would definitely show much higher positive correlation to the *rhythm* characteristic as the lack of it made the samples of very poor quality.

Melody and boringlessness were not that bad compared to the worst characteristic of the three, which was the *euphony*, sometimes considered a major decision maker even through the comments of the respondents, 8.3.1. That is matching our theories that found the low capabilities of the model to grasp chords as a whole, being the biggest flaw even before the actual evaluation by surveys. It also led us to experiment with the proposed GAN architecture and its convolutional chord checker subcomponent.

Variability is the most visible outliner from all the characteristics, being correlated the least (~ 0.10) to each other characteristics. However, the *variablity* is a characteristic featuring the biggest difference between *real* and *generated* samples, where being less variate means a bigger probability of being a *real* sample. That is very interesting and it shows us that the respondents decision making might be highly suboptimal, assigning the lowest decision making priority to the greatest difference. This is also supported by the fact that $\sim 47\%$ of the responses marked the *longrange* samples as real, which we found in Section 8.1.2.

8.2.2 Differences between the results of the main surveys

The two main surveys are *mixed-turing* and *mixed-calibrate*. We have seen rather matching results in terms of rankings relative to each survey, neatly verifying the observed results by bigger number of respondents, in the course of this chapter. How do these results differ in absolute rankings? We proposed and used these two survey versions to see the preconceptions about AI in the light of music generation.

We found that our respondents were much "kinder" in the *mixed-turing* survey on both the real and the generated samples, where the difference for generated samples was bigger. To visualize this tendency, we aggregated characteristics of all the samples, not differing their origins.

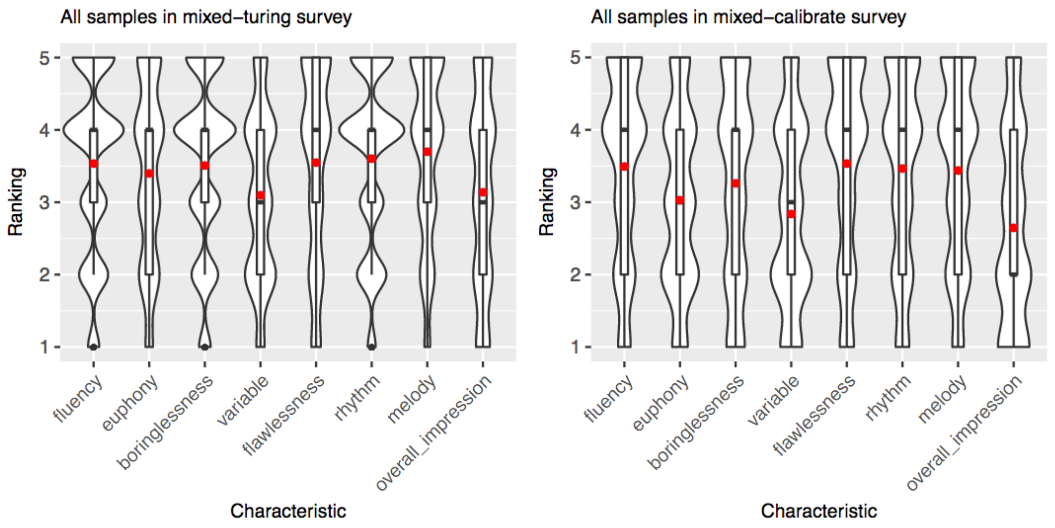


Figure 8.11: Comparison of the two main surveys, which had same samples, but one admitted to contenting a possibility of computer generated samples

Through the aggregation, we found that most differences were rather low and that the generated samples, see Figure 8.5, played bigger part in these differences (real samples were ranked more evenly across the two surveys). In aggregated samples, the biggest differences were in *euphony* and *overall impression*.

The reason for such a difference could be caused by two probable possibilities: bearing in mind that the respondents can bump into generated music causes them to higher their rankings on real/all samples, or, because the *Turing test* question was asked first, they changed their attitude based on their thinking that the sample was generated and bias their rankings with a feeling "good for a computer". From the shown visualizations, we hypothesize that it was a combination of these two, where the second possibility has a bigger role, according to rather low differences seen in Figure 8.11.

8.2.3 Respondents characteristics influence

The respondents were asked about their age, frequency of listening to music, ability to play a musical instrument and their technical and machine learning knowledge. Noting that the number of combinations of these characteristics with the number of music characteristics, number of surveys and origins of the samples is very high, we won't provide any visualizations, other than the attached[12][13] with full (most of the time not interesting) results. Therefore we are going to report only those which were interesting, from our point of view.

The respondents who play a musical instrument enjoyed (*overall impression*) the samples from all the possible origins significantly (and rather equally across origins) more than the others. This trend was the same, but with lower difference for most of the characteristics. They were also more accurate in terms of *Turing test* question.

For the respondents who listen to the music often, we found that they ranked samples of all the possible origins more melodic, euphonic and rhythmic. An interesting observation is that they were much worse in *Turing test/real-likeness* accuracy for both real and generated samples.

The respondents with some technical background or some knowledge about machine learning were also less accurate in guessing the origin of both real and generated samples, and they were also less willing to find some *melody* in all of the samples.

8.3 Qualitative look at the results

Qualitative look through people's comments or authors perspective have a potential to yield or reveal much more than rigid metrics or a rigid selection of survey questions. We added a space for free comments on every sample from the surveys.

[12]All the visualizations are in `surveys` folder, unfortunately with Czech file names. For a better orientation: "nastroj"= musical instrument (plays, doesn't), "posluchacstvi" = listens (often/ not so often)

[13]Some files have bad y-axis labels, use "the higher the better" interpretations as seen throughout this thesis

8.3.1 Most interesting comments

The comments can be divided into those which are of general nature (e.g. most samples are..., the questions are... etc.) and those specific to some sample.

General comments

We discussed results for our survey scheme, not yet with conclusions, in the previous section. To keep in pace, we are going to examine some of the most important views on the survey scheme and samples in general. As the surveys were in Czech language, we will paraphrase the comments.

"The length of the samples is too low to answer anything. In symphonies, there are tons of parts like this. But it has a specific idea, so I think it's human composed"

"The survey was unnecessarily long and it was boring. Appropriate questions would make it more pleasant"

"The difference between a composition and its performance is very unclear"

"A listener who is not interested has nothing to grasp in the samples"

"There was too much samples, it is hard to keep attention so the rankings would be different if I heard it in different order"

"I was able to distinguish only by trying to find some concept within the sample, as there wasn't a problem with anything like melody or rhythm, they just lacked coherence"

"In the samples I marked as generated, there were always some wierd transitions between the chords"

"In the beginning I wasn't able to compare much, therefore I had no idea how big difference between generated and real samples I should expect. The ordering is important because a next sample might have some specific features more or less strong and it can bias the rankings"

We are not going to make any conclusions yet, but all of the comments are very relevant and are similar to our opinions. However they are sometimes in opposition, implying we tried to find some balance. Two of the comments are about the short length of the samples (30 seconds), two of them are about it's high time demands. One is pointing out an unclear division between a composition and its performance, though we pointed out the static tempo, instrument and dynamics in performances, it might be hard to get used to or there may have been some issues with the conversion of real midi files. There were two comments about the ordering and shaping expectations, pointing out the problems we discussed in Sections 6.4.4, 6.4.3. Two last comments were about finding a possible overall feature for a correct distinction between the real and generated samples, those are lacking an overall idea and unusual transitions between the chords.

Comments by origin of the samples

Qualitative comments[14] on the real *classical* samples were often showing a resemble to something. The were also various comments stating the best appealing/rhythmic/mistakelessness etc. With one exception of stating a sample is 90% random.

The *modern* samples were the most commented ones. Frequent comments were about guessing the author, however, they weren't correct most of the time. For example, there were comments guessing the author on a sample, which were from very different eras and distant styles, and no guess was right.

From our perspective, the most important is the qualitative look at the generative models. The *vanilla* samples got the following comments, paraphrased from Czech, (comments of a same sample divided by comma):

"If this is generated, it is great. It distantly reminds me Steve Reich."[15]

"I've unfortunately heard to much things similar to this exactly from its authors. However, I still hope that this is generated. But in fact, the interpretation can work wonders."

"Beethoven?", "Surprisingly this sounds kind of musical. It could be written by a human composer with ease, but with the same ease it could be generated by software", "Alive, interesting, badly performed"

"Too much construed"

"There are some tones omitted"

Paraphrased comments for the second evaluated model, *longrange*:

"There was a feeling of Beethoven", "It sounds like a connection of aleatory and classicism. If this was human composed, it is needed to put someone more aesthetically sensitive into a commission that decides who is a composer and who is not", "Very lively, beautiful passages are alternated by incorrect ones", "It is still the same repeated", "It feels like the composition has blunders and it is too chaotic", "It reminds old baroque music", "If like someone with mediocre musical education tries to imitate Bach for fun", "30% random" - this sample was the best ranked in *Turing test* and *overall impression* out of all generated samples, outranking some real samples as well.

"Sometimes resembled Eben, sometimes Martinů", "In remind music from Scott Joplin", "From atonal compositions, this was the most interesting", "This composition is the first[16] *which I can imagine I do listen to at a concert and I enjoy it."*

[14]File with comments and names of samples bound to them is attached `surveys/surveys_comments`

[15]Steve Reich was not part of the training data.

[16]Sixth in order, preceded by 1 *classical*, 1 *modern*, 1 *longrange* and 2 *vanilla* samples

"Kind of interesting mixture of traditional harmony and something pseudo-modern. However, I cannot guess by 30 seconds long samples", "Also nice"

We can see that while the comments are an interesting and valuable view into the quality of the models, they are very subjective. There were positive and negative comments for all of the samples, real or generated. We've seen that *longrange* samples got more comments than the *vanilla* ones. Some comments on the generated samples were very positive, which is a good result. Comments where the generated samples resembled a human composer were frequent, however, sometimes the authors weren't even part of the training dataset.

We've seen comments on same samples with contradictory claims (e.g. lively, beautiful, still the same repeated, chaotic). It is also hard to average up on comments both for individual samples and for samples of the same origin. We think that comments on individual samples are harder to grasp and that the general comments might have better value for the author. However, comments on individual samples are interesting to read on their own and if they are not obligatory, like in our case, they have a potential to reveal a possible flaw or a good characteristic of a generative model.

8.3.2 Author's impression compared to the results

Since the author chose to evaluate two of the four models, due to time demands of the surveys (average time taken is 30 minutes) and their overhead of processing, we can provide subjective opinions on the other two models. We can also show the subjective, yet very strongly indicating, reasoning behind the selection of the two models for surveys. We also provide opinions on our methods that couldn't be evaluated because of an unclarity on how to do that, or because of the time demands of the surveys.

Majority voting

We used our majority voting improvement 4.3 for both of the models evaluated by surveys. That was done for the sake of not doubling the average time taken responding surveys by evaluating the possible combinations. However, the nature of the improvement doesn't allow automated evaluation because there are two independent models and the output of the algorithm is a binary vector, not suitable for probabilistic losses. Nevertheless, as the author sees it, this improvement help significantly with eliminating mistakes (e.g. out-of-tune notes, notes not timed correctly). For the readers to make their own judgements, see attachments[17].

Baseline model

The author found the results of the **base vanilla LSTM** model lacking a good coherence and information flow through time. We know that a standard RNN

[17] `bad_examples/without_majority_vote.mid`

has these flaws mainly because of the vanishing gradient problems, however the advanced RNN architectures which were designed to capture the long time dependencies have their limits as well (Liu et al. [63]). For the music composition generated by the vanilla LSTM, we can see, that if we extracted two time-distant tracks/passages it would be very hard to see that they are part of one composition.

Proposed advanced models

The other three models feature **skip residual connections** 4.3 in some modification. The model with dense skip connections (*skip-dense lstm*) featured such consistent and coherent information through time that all of the resulting samples sounded very homogeneous and boring[18]. The strength of the information from past time steps was not allowing any interesting variations of consonances/dissonances or shaping of a sensible melody. In theory of Schoenber [11], we've seen that centrifugal forces which are important in music to constitute the potential for development within a musical idea were completely missing.

The *skip-temperature lstm* works the same as *skip-dense lstm* but uses periodically weighted skip connections with a hyperbolic function and a period of one musical bar/measure. Our impression is that while the models has a different architecture than the *vanilla* model, we couldn't tell the possible difference in musical characteristics except we overally enjoyed the results.

Because problems with *vanilla* lacking coherence was more of a long term dependency problem i.e. the model performed very well in terms of time dependencies through e.g. 16 time steps where we couldn't find much difference from real world music. We felt that the subtle transitions and interconnections of passages were crucial, as was also pointed out in comments from the surveys 8.3.1. We added the same periodically weighted skip connections as in *skip-temperature lstm* model, but with three times **longer connections** to help the composition passages carrying a fresh musical idea to stay coherent with the rest of the music. That is, because the information in one musical bar/measure is too low (for most classical compositions) to be efficient for the centripetal essence we would like to capture. The samples from this network are, in our opinion, much more fluent and coherent in terms of general idea and feel less variable, which is also supported by the averaged results from the surveys. We have seen that the real samples in the surveys were less variable too, so it is a good sign.

All the models concluded

For all the evaluated models, we have seen a proneness for mistakes in chords, which is caused by the independent approach to each pitch opposed by astronomically large softmax output layer approach. To compensate this, unlike Lewandowski [40] or Johnson [12] solidly do, we would like to correct the chords by the use of GAN which we outlined but would like to finish it properly in future work. In the most recent days, the independent approach is also opposed by Google Magenta's performance-rnn [38], which features encoding each pitch by a

[18]see attached `bad_examples/skip_dense`

one-hot vector and making time transitions by 100 possible time shift events, also represented by one-hot vectors. Such a network have a high ability to learn most of the chords, but is lacking an overall structure. Moreover, if we represented the musical compositions as performance-rnn did, we would not be able to add the time skip-connections with which we did all the experiments with. That is because of the randomly distributed time domain of the representation and a need for unconditioned architecture of the neural networks.

8.4 What did we learn about the used evaluation scheme

Can we trust the evaluation by surveys? In terms of inter-work comparisons, we've seen that with a sufficient number of respondents (~ 30 in both the main surveys), the two independent surveys showed comparable results and the results also mostly match the author's impression. We didn't try to compare the works of different sources and we think that for this comparison, even the use of the cross-entropy/perplexity loss is not a very reliable indicator for the task of evaluating new creations.

The problem with designing such a survey evaluation is the trade-off between the number of samples, length of samples, number and kind of questions, tediousness of the process of taking the survey, number of possible respondents, characteristics of the respondents, number of calibration listenings, ratio of the real/generated samples and many more. For example, we tried to have the surveys with the lowest possible but also a reasonable amount of samples and length of samples. However, the average time taken was 30 minutes, which is very high for convincing new respondents. We think that if we lowered the number of generated samples or changed the ratio of real/generated, the results will be biased by individual features of some samples and by the shown expectation of half-to-half as shown in Figure 8.1.

We have seen that the diversity of respondents (e.g. plays instrument, listens to music often, has technical background) is very important because of the different overall ratings for these classes shown in Section 8.2.3.

From the chosen questions/characteristics to ask about, we could definitely remove some of them to reduce the time demands of the surveys, we've seen in Figure 8.10, that they are all correlated except the *variability* one, but no one is the most visible candidate and if we take the meanings of the characteristics into account (they aren't almost the same of almost the opposite), it is a difficult task even after we received all the results.

From the results, we can also see that we need to fight the static ordering more, because the samples at the beginning often have unreasonable different ratings from the other samples from the same class. A randomization of the samples for every respondent was already discussed in Section 6.4.3 in the context of number of respondents. We provided two calibration listenings at the beginning of the survey illustrating the range of a possibly met samples. We found that it is needed to add more of these calibration listenings or let them be obligatory.

For more trustfulness we should enlarge the number of samples and for rankings of better quality we should enlarge the length of the samples as well. To compensate these survey length boosts, we should, according to the author, reduce the number of questions to 2 - *Turing test* and *overall impression* with ranking of much higher range e.g. 100, to compensate for the missing questions. In this way, the results could be much better compared with results of works from other sources (although there are still many more variables). However, such results do not help the authors with finding the flaws and and good properties, so we are content about the used evaluation scheme.

8.5 Results of automated evaluation

We also did an automated evaluation with cross-entropy/perplexity loss. The dataset used for this evaluation was different from the dataset for the survey evaluation, which is needed to say because the compexity of the datasets differs. The used dataset was Nottingham 5.0.1, which is well defined in terms of *train, valid and test* parts. Eight variants of our models were evaluated on the test set. We didn't find another work which had both the same representation[19][20], same training data (we omit musical compositions from rhythms other than 4/4) and this kind of evaluation, however, we compare our results to the most basic baseline LSTM model.

We evaluated these variants, described in Section 4.3: *baseline-lstm, skip-dense lstm, skip-temperature lstm, skip-longrange lstm*, all with 0 and 0.2 dropout.

The specific explanation of how the test losses were computed and how the loss was computed is in Section 6.4.2.

Type	**Dropout**	**Test set loss**
baseline-lstm	0.0	0.01659
skip-dense lstm	0.0	0.03451
skip-temperature lstm	0.0	0.01536
skip-longrange lstm	0.0	**0.01492**
baseline-lstm	0.2	0.01614
skip-dense lstm	0.2	0.03464
skip-temperature lstm	0.2	0.01546
skip-longrange lstm	0.2	0.01496

Table 8.6: Averaged binary cross entropy losses bound to 5 best *valid* set losses

For comparison of the results with the results from surveys, the surveys featured samples from *baseline-lstm (vanilla)* and *skip-longrange lstm* with 0 dropout.

We discussed the interpretation issues of the used loss function in Section 6.3. However, if we were to interpret loss on test set like having a lower value means more quality in various characteristics correlated with *"real likeness"* and overall impression, the results seem to match the results of two evaluated models in

[19]Representation is important in evaluation, because it often also defines what do we predict e.g. we predict ligatures/articulations and it is a part of the evaluation

[20]The chosen upper and lower boundaries for *pitch* also affect the evaluation because a number of low and high pitches will be correctly predicted as zeros most of the time

surveys and the other two models from author's subjective view. For a possible match of the reader's impression, see `music_network/all_survey_data` folder attached.

From the versions with dropouts, we can see that the results are very similar in terms of the ordering of the models and also in terms of absolute differences between non-dropout versions. This is a very important observation, because it helps us to verify that these results are not only random numbers.

The best validation losses were found near the end of training so we hypothesize that the chosen number of iterations (same for each network) was suboptimal and we could have got better results. We figured this out in advance, however, for the purposes of convenience with reproducing the results of the 8 trained models we wanted the training to take a reasonably long time.

Conclusion

In this work, we have tried to find new ways of generating music with recurrent neural networks but we also explored other options. We presented an introduction into the problem 1 and its relevant neural network architectures 2.

We described a new modification 4.2 of periodically weighted skip connections to a skip-connected LSTM [6]. To our knowledge, for skip-connected LSTMs, there was no experimental work for music generation. We showed how we experimented with using various frequencies and lengths of skip-connections and evaluated our results. We compared our results to a baseline LSTM model through surveys and showed a solid increase in various characteristics. We found that 47% of our respondents opinions were fooled into thinking our generated samples were coming from a real composer (39% for baseline). We also showed an agreement between survey, subjective and automated evaluation.

We designed a robust evaluation scheme 6.4.2 by surveys featuring four distinct versions of surveys. We also evaluated and presented conclusions on the method itself, on expectations of people and showed preconceptions about AI projected into rankings of the generated samples. Using our survey design, we also closed up on the strong points and the weak points of the evaluated models and how important they are for overall impression.

From not properly finished ideas, we described our findings for generating music in non-standard representations through difference matrices and showed how we tried to cope with them 4.2. We also analyzed possible problems of the related work and outlined interesting and new ideas for coping with them (i.e. use of GAN in "enhancer" mode, design of a discriminator in GAN), which we would like to work on in future work.

Acording to our theories and results, skip-connections in generative recurrent models could be one way of balancing the expectancy and surprise, where the balance is crucial to a specific tension which we like [11] about music. We found that no skip-connections produce rather inconsistent, and too much skip-connections produce very boring samples. The models with "more optimal" amount and length of skip-connections showed stronger results. Therefore, we believe that adding a reasonable amount[21] of skip-connections add to a tension that makes humans perceive music as real or enjoyable.

In future work, we would like to experiment with the newly described representation of discrete music for deep learning by Google Magenta [38], which we would like to combine with the approach of learning the differences of pitches instead of absolute pitches. We would also like to experiment with the idea of training and generating reversed sequences and then reversing them again, because in music, the time dependencies are often bound to the future context.

[21]from possible future experimental or rational advances)

Bibliography

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[2] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[3] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, jan 2016.

[4] Matej Moravcík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael H. Bowling. Deepstack: Expert-level artificial intelligence in no-limit poker. *CoRR*, abs/1701.01724, 2017.

[5] Jurgen Schmidhuber Sepp Hochreiter. Long short-term memory. `http://www.bioinf.jku.at/publications/older/2604.pdf`, 1997. [Online]; Accessed: 2017-04-28].

[6] Yiren Wang and Fei Tian. Recurrent residual learning for sequence classification. In *EMNLP*, 2016.

[7] Theano Development Team. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[8] TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[9] François Chollet. keras. `https://github.com/fchollet/keras`, 2015.

[10] Jan Hajič jr. Handwritten optical music recognition, phd thesis proposal. `http://ufal.mff.cuni.cz/~zabokrtsky/pgs/thesis_proposal/jan-hajic-jr-proposal.pdf`, 2017. [Online]; Acessed: 16-06-2017.

[11] Michael Cherlin. *Schoenberg's Musical Imagination*. Music in the Twentieth Century. Cambridge University Press, 2007.

[12] Daniel Johnson. Composing music with recurrent neural networks. , 2015. [Online]; Acessed: 29-04-2017.

[13] Roger B. Dannenberg. A brief survey of music representation issues,1techniques, and systems. `https://www.cs.cmu.edu/~rbd/papers/issues.pdf`. [Online]; Acessed: 25-06-2017.

[14] Bob L. Sturm Iryna Korshunova, Jo ao Felipe Santos. Folk music style modelling by recurrent neuralnetworks with long short term memory units. `http://ismir2015.uma.es/LBD/LBD13.pdf`. [Online]; Acessed: 25-06-2017.

[15] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546, 2013.

[16] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. Understanding the exploding gradient problem. *CoRR*, abs/1211.5063, 2012.

[17] Christopher Olah. Understanding lstm networks. `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`, 2015. [Online]; Accessed: 2017-04-28].

[18] G.Hinton A.Graves, A.Mohamed. Speech recognition with deep recurrent neural networks. ICASSP, 2013.

[19] Rafal Jozefowicz, Wojciech Zaremba, and Ilya Sutskever. An empirical exploration of recurrent network architectures. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 2342–2350, Lille, France, 07–09 Jul 2015. PMLR.

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[21] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. Residual LSTM: design of a deep recurrent architecture for distant speech recognition. *CoRR*, abs/1701.03360, 2017.

[22] Y. Zhang, G. Chen, D. Yu, K. Yao, S. Khudanpur, and J. Glass. Highway Long Short-Term Memory RNNs for Distant Speech Recognition. *ArXiv e-prints*, October 2015.

[23] CEO Kedar Kulkarni. Convolutional neural networks. `http://hyperverge.co/technology/`. [Online]; Acessed: 6-06-2017.

[24] Yaniv Taigman, Ming Yang, Marc'Aurelio Ranzato, and Lior Wolf. Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 1701–1708, Washington, DC, USA, 2014. IEEE Computer Society.

[25] Pennlio. Fully-connected, locally-connected and shared weights layer in neural networks easy explained. `https://pennlio.wordpress.com/2014/04/11/fully-connected-locally-connected-and-shared-weights-layer-in-neural-netwo` [Online]; Acessed: 6-06-2017.

[26] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122, 2015.

[27] Shuchang Zhou, Jia-Nan Wu, Yuxin Wu, and Xinyu Zhou. Exploiting local structures with the kronecker layer in convolutional networks. *CoRR*, abs/1512.09194, 2015.

[28] Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *CoRR*, abs/1609.03499, 2016.

[29] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Networks. *ArXiv e-prints*, June 2014.

[30] Bing Xu, Naiyan Wang, Tianqi Chen, and Mu Li. Empirical evaluation of rectified activations in convolutional network. *CoRR*, abs/1505.00853, 2015.

[31] Tomasz Talaśka and Rafał Długosz. Initialization mechanism in kohonen neural network implemented in cmos technology.

[32] M. Arjovsky, S. Chintala, and L. Bottou. Wasserstein GAN. *ArXiv e-prints*, January 2017.

[33] Daniel Polani. *Kullback-Leibler Divergence*, pages 1087–1088. Springer New York, New York, NY, 2013.

[34] Alex Irpan. Read-through: Wasserstein gan. `http://www.alexirpan.com/2017/02/22/wasserstein-gan.html`, 2017. [Online]; Acessed: 29-04-2017.

[35] Yuki Inoue Nipun Agarwala and Axel Sly. Music composition using recurrent neural networks. `https://web.stanford.edu/class/cs224n/reports/2762076.pdf`, 2017. [Online]; Acessed: 29-04-2017.

[36] Aran Nayebi and Matt Vitelli. Gruv: Algorithmic music generation usingrecurrent neural networks. `http://cs224d.stanford.edu/reports/NayebiAran.pdf`. [Online]; Acessed: 27-06-2017.

[37] Yoav Zimmerman. A dual classification approach to music language modeling. `http://yoavz.com/music_rnn_paper.pdf`, 2016. [Online]; Acessed: 29-04-2017.

[38] Ian Simon and Sageev Oore. Performance rnn: Generating music with expressive timing and dynamics. `https://magenta.tensorflow.org/performance-rnn`, 2017.

[39] Jurgen Schmidhuber Douglas Eck. A first look at music composition using lstm recurrent neural networks. `http://people.idsia.ch/~juergen/blues/IDSIA-07-02.pdf`. [Online]; Acessed: 29-04-2017.

[40] Y. Bengio N Boulanger-Lewandowski and P. Vincent. 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.

[41] https://magenta.tensorflow.org/, 2016.

[42] Daniel D. Johnson. Generating polyphonic music using tied parallelnetworks. http://www.hexahedria.com/files/2017generatingpolyphonic.pdf. [Online]; Acessed: 11-07-2017.

[43] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[44] D. Randall Wilson and Tony R. Martinez. The general inefficiency of batch training for gradient descent learning. http://axon.cs.byu.edu/papers/Wilson.nn03.batch.pdf, 2016. [Online]; Acessed: 1-05-2017.

[45] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery. *Numerical recipes : the art of scientific computing; 3. ed., this print. is corr. to software version 3.02.* Cambridge Univ. Press, Cambridge [u.a.], 2007. Literaturangaben; English.

[46] KyungHyun Cho, Aaron C. Courville, and Yoshua Bengio. Describing multimedia content using attention-based encoder-decoder networks. *CoRR*, abs/1507.01053, 2015.

[47] Modeling and generating sequences of polyphonic music with the rnn-rbm. http://deeplearning.net/tutorial/rnnrbm.html#rnnrbm. [Online]; Acessed: 03-05-2017.

[48] K. Konda, R. Memisevic, and D. Krueger. Zero-bias autoencoders and the benefits of co-adapting features. *ArXiv e-prints*, February 2014.

[49] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[50] Andrew Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04, pages 78–, New York, NY, USA, 2004. ACM.

[51] Bernd Krueger. Classical piano midi page. http://www.piano-midi.de/. [Online]; Acessed: 25-06-2017.

[52] James Allwright. Nottingham music database. http://abc.sourceforge.net/NMD/. [Online]; Acessed: 25-06-2017.

[53] CCARH Walter Hewlett. Musedata. http://www.musedata.org/. [Online]; Acessed: 25-06-2017.

[54] Center for computer assisted research in the humanities at stanford university. http://wiki.ccarh.org/wiki/Contact_information. [Online]; Acessed: 5-06-2017.

[55] e-piano junior competition. http://www.piano-e-competition.com/. [Online]; Acessed: 11-07-2017.

[56] I-Ting Liu and Richard Randall. Predicting missing music components with bidirectional long short-term memory neural networks. `www.andrew.cmu.edu/user/randall/publications/BLSTM_ISMIR16_FINAL.pdf`, 2016. [Online]; Acessed: 6-06-2017.

[57] Daniel D. Johnson. *Generating Polyphonic Music Using Tied Parallel Networks*, pages 128–143. Springer International Publishing, Cham, 2017.

[58] Richard S. Sutton and Andrew G. Barto. Reinforcement learning:an introduction. `http://incompleteideas.net/sutton/book/bookdraft2017june19.pdf`, Draft, 2014, 2015, 2016, 2017. [Online]; Acessed: 6-06-2017.

[59] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[60] Rishabh Shukla. http://rishy.github.io/ml/2017/01/05/how-to-train-your-dnn/.

[61] Keunwoo Choi, George Fazekas, and Mark B. Sandler. Text-based LSTM networks for automatic music composition. *CoRR*, abs/1604.05358, 2016.

[62] Matthew D. Zeiler. ADADELTA: an adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[63] Pengfei Liu, Xipeng Qiu, Xinchi Chen, Shiyu Wu, and Xuanjing Huang. Multi-timescale long short-term memory neural network for modelling sentences and documents. In *EMNLP*, 2015.

List of Figures

[22]We don't use boxplots exclusively because of the small scale of ratings where the sections bounded by quartiles are often too large.

List of Tables