

Detecting Musical Rhetoric Figures with LSTM using Procedurally Generated Synthetic Data

Niek de Gier
4136853

Master Artificial Intelligence
Utrecht University

Master Thesis

Supervised by
dr. ir. P. (Peter) van Kranenburg

Second examiner
dr. F. (Frans) Wiering

July 3, 2022

Abstract

Musicologists have researched rhetorical techniques applied to baroque music, which includes the works of Johann Sebastian Bach. These musical rhetoric figures come in the form of rhythmic and melodic patterns, with each figure having the goal of evoking a certain emotion or Christian symbolism. This thesis presents a machine learning approach to pattern recognition in symbolic music. A Long Short-Term Memory (LSTM) model will be trained to detect the figura corta, a rhythmical figure, in the cantatas of J.S. Bach. Since a labeled dataset of Bach's cantatas does not exist and labeling the data manually would be extremely time consuming, this thesis approaches the problem by generating a dataset from scratch. By drawing up laws and rules to which the data should abide, an algorithm is presented that generates plausible musical fragments as training input based on these criteria. Then, twelve different parameter settings were explored by training an LSTM on the resulting datasets. The best performing model was subsequently tested on six real cantata movements. On average, the LSTM model achieved a precision of 78.53%, a recall of 83.12% and an accuracy of 94.46%. From these results it is concluded that synthetic data can produce a reliable dataset to train an LSTM that can successfully classify real data on whether it contains a figura corta or not. Furthermore, this result implies that the presented method of procedurally generating data produces a varied and correct dataset. By extension, it implies that the laws and rules proposed, as well as the representation of the musical data, allow the LSTM to correctly apply its learned features to real data.

Acknowledgement

I would like to thank Peter van Kranenburg for the weekly meetings I had with him. These meetings, in which we discussed my work and methods, have provided me with the necessary perspective to keep improving and going forward.

I also would like to thank Frans Wiering for the valuable feedback on my research proposal. While the plan shifted away from that proposal in the past 6 months, your critical remarks still helped me when writing this final version.

Contents

1	Introduction	1
2	Related Work	4
2.1	Musical Rhetoric Figures	4
2.1.1	Musical rhetoric in the baroque period	4
2.1.2	Figura Corta	5
2.1.3	Bach’s cantatas	6
2.2	Music Information Retrieval	6
2.2.1	Pattern recognition within MIR	6
2.3	Long Short-Term Memory	7
2.4	Synthetic data	8
2.4.1	Synthetic data in Computer Vision	9
2.4.2	Synthetic data in Neural Programming	9
2.4.3	Synthetic data in MIR	10
2.4.4	Procedural data generation	10
3	Methods	12
3.1	Synthetic data	12
3.1.1	Criteria	12
3.1.2	Generating data	14
3.1.3	Data representation	16
3.2	Detecting the figura corta	18
3.2.1	LSTM architecture	18
3.2.2	Parameters	18
3.3	Evaluation	19
4	Results	21
4.1	Accuracy on synthetic test data	21
4.2	Results on real data	22
4.2.1	Precision	22
4.2.2	Recall	22
5	Conclusion	26
6	Discussion	27
6.1	Realism of synthetic data	27
6.2	Error analysis and improvements	28
6.3	Future work	29
6.3.1	Using GAN for synthetic data	29
6.3.2	Detecting other figures	29

6.3.3	Analysis of musical rhetoric	30
References		31

Chapter 1

Introduction

In the early Renaissance, large amounts of ancient Roman and Greek texts were recovered. The works of Greek philosophers have made their mark on many aspects in life at the time, especially in arts and rhetoric. Connections between oratory and musical composition have been suggested since the Renaissance (Couch, 1984), and research has been done by musicologists in search of rhetorical techniques applied to music, especially in sixteenth to eighteenth century western baroque music (Benitez Jr, 1985; Couch, 1984; Van Dijk et al., 1981), which includes the works of Johann Sebastian Bach. These musical rhetoric figures are hypothesized to occur frequently in music written during the baroque period in Western Europe. They come in the form of rhythmic and melodic patterns, with each figure having the goal of evoking a certain emotion or Christian symbolism (Van Dijk et al., 1981). Many of these figures have been described in *Musikalisches Lexikon* (Walther, 1732) and *Musica Poetica* (Bartel, 1997). Bach has composed over one thousand works, as seen in the Bach-Werke-Verzeichnis (BWV) (Schmieder, 1950), an extensive catalogue listing all of Bach’s known compositions. This fact makes the manual search and analysis of rhetorical figures only feasible in small subsets of Bach’s repertoire. We envision that the detection and analysis of these figures can be done using machine learning in order to support the ongoing research by increasing the speed and enlarging the scope of the analysis. This combination of artificial intelligence and musicology makes it a part of Music Information Retrieval (MIR).

MIR is a growing field of research centered around the extraction of meaningful features in music. MIR has a broad range of applications and topics, including genre classification, audio recognition and automated music composition. Additionally, a number of researchers in the field of MIR have been looking to automatically detect patterns within symbolic music. Lartillot (2005, 2014) used pattern recognition to find (cyclic) repetitions with possible variations, and several authors have published algorithms and models to detect melodic patterns between multiple voices (Finkensiep et al., 2020; Katsiavalos et al., 2019; Symons, 2017).

In this thesis, we will take a machine learning approach to pattern recognition in symbolic music, i.e. written music or sheet music. More specifically, by applying a neural network architecture that is prevalent in Natural Language Processing (NLP), we will aim to detect the figura corta, a rhythmical figure, in the cantatas of J.S. Bach. This architecture, the Long Short-Term Memory (LSTM), is specifically designed to analyse sequences (in NLP, these are usually sentences) by ”remembering” and connecting different parts of the sequence. In NLP, context matters a lot: the beginning of a sentence can have an important connection to the end of the sentence. The LSTM, while not the first architecture of its kind, is effective in understanding these connections even in larger

sequences (Yu et al., 2019). It is not hard to find a parallel between NLP and symbolic music: both text and music are made up of individual elements that together form a larger sequence. Words combine to form a sentence, just like notes combine to form a musical phrase. And just like in NLP, there are important contextual dependencies to find between different parts of a music fragment. In short, we hypothesize that an LSTM model should be able to extract meaning from music just as well as it can extract meaning from text. In this thesis, the pattern we will be looking for is the *figura corta*, one of the many musical rhetoric figures present in Walther's *Musikalisches Lexikon*.

One glaring issue with using an LSTM architecture lies in the data. The LSTM model is a supervised machine learning model, meaning that the data it trains on should have a label on every data point saying whether the musical fragment contains a *figura corta* or not. While the *figura corta* has been found in Bach's cantatas (Van Kranenburg, 2010), there is no labeled dataset available. We will manually annotate some cantata movements to have some form of ground truth, but manually annotating all cantatas - each with multiple movements - in search of the *figura corta* would be extremely time consuming, not to mention that this would also need to be done for every other figure we want to detect in future research. Instead, we will approach this problem by generating our own dataset. This concept of synthetic data is already being applied in different fields within artificial intelligence. Wang et al. (2019) rendered fake images of crowds with specified amounts of people in them in order to train a model that can reliably estimate the amount of people in real-life crowds. In a paper more closely related to NLP, Gupta et al. (2016) designed an engine that overlays text on top of images, which were then used as training data for a model that detects text in real images.

To bring it all together, we will design an algorithm that procedurally generates musical fragments which can be injected with the *figura corta*. This algorithm will provide us with a labeled dataset in a vectorized representation fit for training an LSTM model. This LSTM model will in turn be tested on the actual cantatas by Bach to see how well the learned features from the synthetic data can be transferred onto the real data. This approach will answer the following research question:

*Is it possible to reliably detect the *figura corta* in Bach's cantatas with an LSTM by training the model on a synthetic dataset?*

To answer this question, we first need to design an algorithm that generates the dataset. This results in a few sub-questions:

- 1.1 *Which criteria does the dataset need to satisfy in order for the LSTM to apply its learned features to real data?*
- 1.2 *How do we introduce enough variety to the dataset to allow the LSTM to generalize well?*
- 1.3 *In which way should we represent the data in order to expose relevant musical features to the LSTM?*

When we have our dataset, we need to train and test the LSTM on this synthetic data in order to find a good candidate model to test on the real cantatas. This raises the following questions:

- 2.1 *How do we turn an LSTM into a suitable binary classifier so that it can detect the *figura corta*?*
- 2.2 *What set of parameters will we vary in order to find the best resulting LSTM model?*

2.3 *How do we evaluate each trained LSTM model on its potential to transfer its learned features to real data?*

After evaluating different parameter combinations, our best LSTM model will be used to test on real cantatas. In order to finally answer our original research question, we ask ourselves one more question:

3.1 *How do we evaluate whether the trained LSTM can reliably detect the figura corta in Bach's cantatas?*

If this thesis produces desirable results, it opens up further research into the possibility of using synthetic data in this way. While simpler pattern matching approaches might be a more straightforward method to finding the figura corta, it is also a more manual and specific way to find it. For example, by using a regular expression, finding simple figures can be an almost trivial task. However, when trying to detect multiple different figures, each with different variations and possible ornamental notes, the amount of hand-crafted pattern matching algorithms required would grow exponentially. Contrast this with generating the figures: by using stochastic processes one can easily introduce random noise and variations to the dataset, which a neural network could learn to recognize. Also, the representation of the data can stay mostly the same, or can be extended without losing the ability to detect other figures. The automatic localization of the figura corta can support further analysis into the distribution of the figure, as well as further research into the sung lyrics that are accompanied by the figura corta, in order to infer the goal of the rhetorical technique behind the figure.

Chapter 2

Related Work

In this chapter, we will touch upon the research fields related to this thesis. The motivation for our research question stems from musicology, specifically the research with respect to musical rhetoric figures in the baroque period of Western Europe. We will give a context for this field in section 2.1 and the primary object of this thesis: the *figura corta*.

Given the use of artificial intelligence and computational musicology in our approach to answering our research question, we will then give an overview of Music Information Retrieval. Detailed in section 2.2, MIR is a field of research in which computational and artificial intelligence is used to extract and analyse meaningful features from musical data.

In section 2.3, we will provide an introduction into the Long Short-Term Memory architecture. This neural network architecture will be the base for the models we will train, and belongs to the class of Recurrent Neural Networks (RNN), a type of neural network often used to deal with sequential and temporal data, in this case symbolic music.

Lastly, in section 2.4 we will give an impression of the use of synthetic data within artificial intelligence. Synthetic data is constructed data, either by hand or automated, that is used in place of, or in conjunction with, real data. The use of synthetic data can be out of necessity when there is no labeled data available. In our approach to using supervised neural networks to detect the *figura corta*, we run into that exact problem. However, we believe that synthetic data is not just a necessity for us, but that it will actually provide us with the means to quickly and reliably generate more data than would be feasible when manually labeling existing music.

2.1 Musical Rhetoric Figures

We use the term *musical rhetoric figure* for the patterns and motifs used by sixteenth to eighteenth century Western European composers to evoke certain emotions in the listener. In this section we will briefly describe the research into these figures, as well as highlight one the *figura corta*.

2.1.1 Musical rhetoric in the baroque period

During the baroque period of Western Europe, from the sixteenth to eighteenth century, the art of rhetoric was used in all forms of art, including music. To quote Van Dijk et al. (1981): "Componeren was in die tijd als het schrijven van een redevoering, musiceren als het voordragen ervan." ("In that time, composing was like writing a speech, performing was like reciting it."). Instrumental music was considered a language, meant to persuade the listener. This rhetoric stems from ancient Greece, where philosophers and

writers developed techniques to inspire and persuade when speaking. The use of rhetoric in the baroque period directly tied into the *Affektenlehre*: the study of emotions. Composers were trying to find ways to invoke certain subjective emotions (like joy or anger) in the listener using these rhetoric techniques. (Wessel, 1955). In "Musiceren als Brugman", Van Dijk et al. (1981) describe how the five pillars of classical rhetoric were used to compose and perform a musical piece in the baroque period. Starting with *Inventio* (invention), the composer decided upon the themes of the piece, mostly guided by the lyrical component of the piece. In baroque Germany, these themes were often Christian, the texts originating from the Lutheran bible. Then, the composer had to create an outline for the piece, using the second pillar of rhetoric, *Dispositio* (arrangement): Starting with an introduction to get the attention of the listener and exposing the important motifs, then getting to the core by expanding on those motifs and providing countermelodies (as a way of argument and counterargument), and finally concluding the piece by recapitalizing on the original motifs. The third pillar of rhetoric is *Elaboratio* (elaboration): this is where the musical rhetoric figures come into play. Like a speaker using idioms and figures of speech, a baroque composer could use predefined patterns and motifs to embellish their melodies and themes. Many of these figures have been described and analysed over the course of centuries Bartel (1997), Dings (2022), Van Dijk et al. (1981), and Walther (1732). These figures were often used as support for the lyrics, and as such, have a certain emotion or deeper meaning tied to them. The last two pillars, *Memoria* (memory) and *Actio* (performance), were for memorization of the composition and how to perform it. While *Memoria* was important in the original Greek rhetoric, there is no indication that it was equally as important when performing baroque music (Van Dijk et al., 1981). The performance of the piece was important, however, as the emotions invoked in the listener had to be evident in the performer as well, i.e. using body language and intonation. In this thesis, we will focus on the musical rhetorical figures of the *Elaboratio*, specifically the *figura corta*.

2.1.2 Figura Corta

The *figura corta* is described as a rhythmical figure consisting of groups of three notes, where one note is twice as long as the other two (Walther, 1732). This gives three possible variations of the figure: short-short-long, long-short-short and short-long-short, although the latter is not often considered in the literature. For example, Benitez Jr (1985) only mentions the first two variants, being *anapest* (short-short-long) and *dactyl* (long-short-short). These terms are directly related to the anapest and dactyl in poetic meter, where they are expressed through stressed (long) and unstressed (short) syllables (Kiparsky & Youmans, 2014). One well known example of a dactyl in poetry is the limerick: apart from the first syllable (which can be seen as an *anacrusis*¹), the accents of the limerick follow the pattern of stressed-unstressed-unstressed. One thing to note here is that the way in which anapest and dactyl are used in poetry has not necessarily to do with rhythm. Stress in spoken language can be achieved through intonation alone, but the musical *figura corta* is only described in terms of note length, not accents. The *figura corta* signifies joy and happiness and can portray the faith in God, according to Van Dijk et al. (1981). However, they did not provide much argumentation for this conclusion. Schweitzer (1908) names the repeated dactyl in Bach's work as the "Freudenmotiv" ("joy motif") (Van Kranenburg, 2010), which does strengthen the claim that the *figura corta* is used as a way to invoke

¹An anacrusis in poetry and music is a set of introductory words or notes that occur before the actual start of a phrase.

joy and happiness in the listener. However, there is still no substantial evidence to prove these claims. In order to be able to infer meaning behind the *figura corta*, Van Kranenburg (2010) proposes to analyse lyrics that accompany the *figura corta*. The first step in the process of getting substantial empirical evidence is to find lots of occurrences of the figure in Bach's work. This is the primary motivation for the methods we present in this thesis, as we wish to automate this process.

2.1.3 Bach's cantatas

We will specifically look for the *figura corta* in Bach's cantatas. A cantata (conjugation of the Italian verb *cantare*, "to sing") is, in its basic form, a sung piece (often by a choir) with instrumental accompaniment. The cantatas composed by Bach are mostly comprised of a 4-part choir with Bass, Tenor, Alto and Soprano accompanied by a variety of instruments including harpsichord, organ, stringed instruments and at times a woodwind section. The lyrics are in large part German protestant church texts, often adapted from the works of Martin Luther. We have a large collection of digitalised cantatas available².

2.2 Music Information Retrieval

In this section we will provide a short overview of the broad field of MIR, after which we will zoom in on the aspects of pattern recognition within MIR to which this thesis closely relates.

Music Information Retrieval is an interdisciplinary research field dedicated to extracting and analysing meaningful features from musical data, be it audio, symbolic (written) music or metadata like tags that describe particular genres (Futrelle & Downie, 2002). While the goal of the field was to "making the entire corpus of music readily accessible" (Downie, 2003), it has attracted a whole range of applications. Being a part of Information Retrieval, a big part of MIR involves retrieving correct pieces of music depending on a query. This includes well-known systems like Spotify (recommending music based on perceived interest) and Shazam (recognizing and retrieving songs by audio input). These applications are a manifest of what Schedl et al. (2014) describes as "a shift away from system-centric towards user-centric designs". The commercial application of MIR is one of the reasons of the growth of the international MIR research community (Downie, 2003). Other subfields in MIR include Optical Music Recognition (OMR, detecting symbolic music in images) and pattern recognition (finding musical patterns within symbolic music). In this thesis, we will focus on the pattern recognition aspect of MIR, given the nature of the musical rhetoric figures.

2.2.1 Pattern recognition within MIR

A large part of MIR encompasses finding and analysing patterns within symbolic music. This area of computational musicology can be divided into two disciplines: Pattern discovery and pattern matching. Pattern discovery can be viewed as the *unsupervised* search of patterns within symbolic music that are not necessarily known beforehand, while pattern matching (or localization) is the often *supervised* task of finding predefined patterns (Janssen et al., 2013; Rolland, 1999). While closely related, the methods may differ vastly between the two.

²125 cantatas are digitalised in the MuseData (.md) format by the Center for Computer Assisted Research in the Humanities, found on <http://esf.ccarh.org/MuseData-Ed-202003/baroque/bach/bg/cant/>

A pure pattern discovery approach was conducted by Lartillot (2005). In this paper, Lartillot proposed a sequence-based pattern discovery algorithm to find closed motivic patterns within symbolic music. Closed motivic patterns were defined as fragments that are repeated multiple times in a single piece of music, albeit with varying pitch or rhythm, as long as a part of the structure remains the same. This can mean a rhythmic pattern that varies in pitch, or a melodic pattern that preserves its intervals. As a data representation, Lartillot used pitch and rhythm features to describe the change between consecutive notes. While the definition of a closed motivic pattern is close to a rhetorical figure in the sense that the figures can vary in the same way, the patterns Lartillot was looking for were not predefined. Instead, these patterns were confined within a single musical piece, and were not necessarily overarching patterns used in other pieces as well.

Giraud et al. (2012) utilized an unsupervised pattern matching approach to find occurrences of a fugue’s subject throughout the rest of the fugue. While not necessarily a predefined motif, Giraud et al. used the fact that in Bach’s *Well-Tempered Clavier*, the subject of the fugue is always isolated in one voice at the start. Since the end of the subject is not always clear, they tried to match different subject candidates against the rest of the voices to see if they all have that same ending point. Afterwards, they searched the whole fugue for the pitch intervals that were present in the first subject in order to find similar matches.

As a research topic closer to ours, Finkensiep et al. (2020) set out to detect voice-leading schemas. These schemas are, like figures, predefined and thus the method involves pattern matching. Voice-leading schemas are defined by Finkensiep as ”configurations of two or more voices that move together through a sequence of stages, forming particular patterns of successive vertical intervals that occur within a specific tonal context”. The schemas are composition techniques that function as standardized ways to introduce certain harmonic context to the music, mostly through tension and release. The detection of these schemas was done using logistic regression, a machine learning technique that uses a basic neural network to classify the input.

2.3 Long Short-Term Memory

The neural network architecture we will be using in this thesis is the Long Short-Term Memory (LSTM) network. LSTM is a type of Recurrent Neural Network (RNN), a class of neural networks that work with sequential data and are made to learn time-varying patterns (Medsker & Jain, 2001). An RNN has continuous-valued states, with each state being a function of the previous one (Rodriguez et al., 1999). This means that for every input element in the sequence, an RNN will consider previous input that it has already seen in order to produce its current state, hence the recursion. The RNN has seen great success in NLP tasks like machine translation and text generation, but when the gap between time steps that are contextually dependent becomes larger, standard RNNs become unable to learn these long-term dependencies (Zhou et al., 2015). In order to solve this problem, the Long Short-Term Memory (LSTM) architecture was introduced by Hochreiter and Schmidhuber (1997) and expanded upon by Gers et al. (2000). With its input, output and forget gates, the LSTM network can more easily detect dependencies between different parts of the input. This is particularly helpful in NLP, since sentence structure is often independent of its content. This means that two words that are semantically or syntactically connected can be far apart, and LSTM can pick up on those dependencies.

Like a standard RNN, the LSTM architecture consists of repeated cells for every time step in the input sequence. In an LSTM architecture, the inner workings of each cell

regulate the flow of information using a number of *gates*. In their first iteration of the LSTM, Hochreiter and Schmidhuber implemented cells consisting of the cell state s_t , the cell input activation c_t and a hidden state h_t . This hidden state may also be used as an intermediate output while processing the input sequence. They also added an input gate i_t and output gate o_t , regulating the amount of information going respectively in and out of the cell. In 2000, Gers et al. introduced the forget gate f_t , which acts as a means to gradually forget information from previous cells when that information is no longer relevant. Each gate (i_t , o_t and f_t , as well as the cell input activation c_t), has a similar internal structure. This structure consists of a bias b and two weight matrices W and U , which are all shared between time steps and updated during training by back propagation. The bias acts as a preliminary belief about the particular gate and is added independent of input or previous hidden state. The weight matrix W acts on the current input, and the weight matrix U acts on the hidden state (or output) of the previous cell. Together, these three trainable elements form the rules for each gate, telling the cell what to focus on when considering the input and previous hidden state, before adding the independent bias. After adding the results, an activation function is applied. The three gates and the cell input activation all use the sigmoid activation function $\sigma(z) = \frac{1}{1+e^{-x}}$ to get the resulting value between 0 and 1. Each cell then computes its cell state s_t by first multiplying the forget gate with the previous cell state - you can think of this as selectively forgetting and remembering previous information - and then adding the input gate multiplied with the cell input activation. This cell state will be used again in the next cell, as well as when computing the current hidden state h_t . This hidden state, or intermediate output, is calculated by activating the cell state with the hyperbolic tangent (to squeeze its value between -1 and 1) and multiplying it with the output gate to regulate the flow of information going out of the cell. To put it all together:

$$\begin{aligned}i_t &= \sigma(W_i x_t + U_i h_{t-1} + b_i) \\c_t &= \sigma(W_c x_t + U_c h_{t-1} + b_c) \\f_t &= \sigma(W_f x_t + U_f h_{t-1} + b_f) \\o_t &= \sigma(W_o x_t + U_o h_{t-1} + b_o) \\s_t &= f_t \circ s_{t-1} + i_t \circ c_t \\h_t &= o_t \circ \tanh(s_t)\end{aligned}$$

with the operator \circ being element-wise multiplication.

The hidden state of the last cell is considered the output of the LSTM, and can be run through a softmax layer to transform it into a probability distribution. Depending on the goal, this output can be used as a prediction for the next element in the sequence, or as a classifier over the input. The LSTM network is a supervised machine learning model and is trained by backpropagation through time using an optimization algorithm (like Stochastic Gradient Descent or Adam) in order to update its weights.

2.4 Synthetic data

In this section we will give a general overview of the use of synthetic data within Artificial Intelligence. Synthetic data is a quickly growing approach described by Nikolenko (2021) as "producing artificial data from scratch or using advanced data manipulation techniques to produce novel and diverse training examples", in order to generate a dataset for a problem that does not have sufficient or acceptable training data. This definition

distinguishes synthetic data from more standard data manipulation and augmentation. For example, in an image recognition problem, data augmentation may include taking existing labeled images, and performing geometric transformations like flipping or rotating the image (Shorten & Khoshgoftaar, 2019). This preserves the existing label, while creating new training input. Synthetic data takes this concept a step further by producing *novel* data, even when there is no existing data to work with. Sometimes this can mean using existing data to create something new, like pasting a picture of a cat on top of the background of another image, which may be referred to as "smart augmentation", a grey area between synthetic data and data augmentation (Nikolenko, 2021).

2.4.1 Synthetic data in Computer Vision

Most applications of synthetic data are found in Computer Vision, the field of research within AI dedicated to analysing, classifying and generating visual input. Originally, synthetic data was mostly used for evaluation, for example in optical flow estimation (the estimation of object movement through time) (Nikolenko, 2021). More direct uses of synthetic data are found in Optical Character Recognition (OCR), the detection and recognition of text in images. One such dataset involved automatic generation of textual characters by using computer fonts (De Campos et al., 2009). In a more high level computer vision task, Wang et al. (2019) sought to estimate the amount of people present in a crowd. They used a graphics engine to produce synthetic data by rendering a large corpus of images containing a predefined number of people. This way of generating a synthetic dataset naturally allowed the machine learning model to transfer its learned features to real data. Although the field of Computer Vision is not related to pattern recognition within MIR, this concept of transferability will be important in our research: since the model can only learn from constructed data, the data must be representative of the real data so that the model can learn actually useful features and recognize them in a real dataset.

2.4.2 Synthetic data in Neural Programming

Different areas within AI are starting to develop their own approach for synthetic data. In neural programming, the field consisting of automated programming using machine learning, synthetic data is used by necessity, since there is no sufficiently large dataset of programs with their respective input and output to train on. Zaremba and Sutskever (2014) designed an algorithm that can produce a complete dataset of simple programs (that can be run in linear time) with a corresponding target output. This dataset was used as training data for an LSTM with the goal of predicting the output given the input program, which was successful for programs involving simple addition and memorization. Shin et al. (2019) assessed the state of current approaches to generating synthetic data in neural programming. After concluding that these approaches may unintentionally create bias in the datasets, which prevented the models from generalizing well, Shin et al. introduced criteria or desired features to the data generators, allowing them to manipulate the distribution of the resulting dataset. The models that trained on these datasets showed an improvement in their ability to generalize. This approach is viable when the scientists involved have the domain knowledge necessary to steer the data generator in the right direction without loss of generalization. In this thesis, we will apply this line of thinking by defining constraints that our data generator must adhere to.

2.4.3 Synthetic data in MIR

Cífka et al. (2019) trained a model based on a sequence-to-sequence RNN architecture with the goal of transforming musical fragments to other styles, while retaining the general structure of the notes. For example, when given a 12-bar blues and a desired output style, e.g. samba, it should produce a musical fragment in the samba style that follows the same melodic line as the input. This mostly involves the rhythmic patterns unique to each style. The dataset used in this paper was produced by using the Band-In-A-Box (BIAB) software³. This software creates musical fragments in MIDI format by taking a series of chords as input. Cífka et al. used the BIAB software to create pairs of input/output by supplying the software with a chord progression and generating two music fragments in the input and output style respectively. This synthetic dataset was then successfully used to train the models, which could then produce "musically meaningful accompaniments".

Choi et al. (2018) used synthetic data in an Optical Music Recognition (OMR) task, where the goal was to reliably detect symbolic music in images, especially old manuscripts where the images may be heavily degraded. This task is closely related to Computer Vision, and their approach to synthetic data also reflects this. Training data was created using the Musescore software⁴, which has the ability to generate symbolic music together with an annotation in the form of an XML file stating which symbols are used. The images containing the symbolic music were then altered by adding noise and deformation, and augmented using simple data augmentation like flipping, rotating and translating the images. There were no results presented in this paper, and they did not seem to pursue this method in further research (Choi et al., 2019).

All in all, there have not been many uses of synthetic data in Music Information Retrieval as of now. The current approaches mostly consist of manually crafting a synthetic dataset, albeit with the help of some form of automation. In contrast, our approach will lean more into *procedural data generation*.

2.4.4 Procedural data generation

Nikolenko (2021) discusses promising future work at the end of their book. One technique they highlighted was procedural data generation, an approach in which an algorithm can generate a theoretically infinite amount of data by randomly selecting different features that will together form a realistic data point. Qi et al. (2018) have presented a method in which they generate random room layouts that can be used as synthetic data for a variety of computer vision problems, including image recognition and 3D content generation. In their paper, Qi et al. apply various stochastic processes in order to generate indoor scenes, determined by the human interaction within the room. Their approach involves sampling layouts from real indoor scenes, Markov Random Fields for the human context and a probabilistic grammar model to bring it all together. By comparing their generated data to existing data, they concluded that their data achieved higher realism than other methods. On top of that, they asked human subjects to rate their generated rooms according to functionality and naturalness. While this achieved high scores with an average of 4.108 out of 5 for functionality and an average of 3.884 out of 5 for naturalness, they only approached 4 human subjects to evaluate their data. Because of this, we treat these results as a good indication, but not as conclusive proof. Because 3D content generation is an especially complex problem, the approach of Qi et al. is a bit excessive for generating short music fragments like we intend to do in this thesis. However, we will

³<https://www.pgmusic.com/>

⁴<http://www.musescore.org>

follow the general idea of this approach by determining criteria for our data and finding suitable stochastic processes that will generate a diverse and non-trivial dataset.

Chapter 3

Methods

In this chapter we will describe our method to train an LSTM model on synthetic data with the aim of detecting the *figura corta* in Bach's cantatas. In section 3.1, we will construct an algorithm that procedurally generates fragments of music, which will be used to generate an entire dataset that acts as training and validation input for an LSTM model. This LSTM model will be detailed in section 3.2, and will have the goal of detecting the *figura corta*. Further in this section, we will explore different parameter settings for the algorithm and the LSTM model. In section 3.3 we describe how we will determine which of these parameters settings produces the best results when detecting the *figura corta*. Also in this section, we will present how the resulting model will be tested further on the actual cantatas to evaluate whether the learned features are transferable to real data.

3.1 Synthetic data

In this section we will first provide a set of criteria to which the synthetic data should adhere, based on the structure of music written during the baroque period of western Europe. This will propose an answer to research question 1.1. We will then design an algorithm that will satisfy these constraints using a mix of randomness and predetermined patterns, answering research question 1.2. Lastly, to answer research question 1.3, the data will be represented in a way that describes the symbolic music in terms of its important features.

3.1.1 Criteria

In order to generate realistic data, we need to come up with a set of guidelines that the algorithm must follow. We will split these criteria in two categories, based on the work of Meyer (1996): laws and rules. Laws are defined as physical or psychological constraints in the music, and they will prevent the randomness in the data from creating absurd fragments (e.g. fragments containing notes that can not be played on the instruments used), while rules are based on the musical style of the music, in this case Baroque cantatas. These rules will mold the fragments into a pattern-based structure reminiscent of Bach.

The laws we define are as follows:

1. **Notes cannot go lower or higher than the lowest and highest note in the cantata dataset respectively.** This ensures that all notes can actually be played by at least one instrument from the baroque period.

2. **Notes and rests must have a duration of no less than 1/32 of a whole note, and no more than 2 whole notes.** While the cantatas do contain note durations outside this range, they are very sparsely used, so this range represents the reality without over-complicating the duration space.
3. **Notes and rests must have a duration that is present at least once in the cantatas.** This prevents anomalies like complicated fractions or even irrational durations. While fractional durations are present in the data, they are mostly simple fractions representing triplets, quintuplets, etc.

Using these laws, we have defined the domain of our notes and rests based on the cantatas. Obviously these constraints are not enough to consistently output realistic fragments, so we need a set of soft, pattern-based rules that will make the fragments melodically and rhythmically plausible.

1. **All notes in the fragment must be part of the same key and scale.** In the baroque period, music was mainly written using scales, which are defined by a *key* and a mode. The key acts as the base note, or tonic, for the scale. The mode determines which notes can be played in between the tonic and its next occurrence, one octave up. For example, if the algorithm generates a sentence using the D major scale, all notes in that sentence must be part of the set $\{D, E, F\sharp, G, A, B, C\sharp\}$, but the octave in which they are played may vary.
2. **The algorithm must prefer small intervals between consecutive pitches.** After examining Bach's cantatas, we noticed that lots of melodic lines are comprised of small intervals between the pitches. Larger intervals are found more sporadically, but can definitely be used as well. This leads to the rule that, while every interval technically can be used, the algorithm should favour smaller intervals with a quickly decreasing chance for larger intervals.
3. **The algorithm must assign durations to notes and rests based on a realistic pattern.** This one is the hardest to define, but assessing the cantatas, we see that 1/8 and 1/16 notes are most prevalent. Also, the rhythmical structure generally does not change multiple times in a small time frame. I.e., a sequence of half notes followed by a sixteenth and then back to a half note is rare and sounds unrealistic. Since it is very hard to precisely define the criteria for duration patterns, we will try three different methods, which will be described in the next section.
4. **There must be a logical variety of notes and rests.** There are no real rules for the distribution between rests and notes, so we will experimentally create a realistic distribution in the next section.
5. **Sentences containing figures must lead and end with 0 or more words.** As we have observed in the data, figures do not exclusively appear at the start of a measure. This rule makes it possible for a figure to appear at any place in a fragment, making it appear as part of a larger piece of music. Furthermore, it forces the model to learn the pattern of the figure, since it will not be able to overfit on the location of the figure inside the fragments.

When the output of the algorithm meets these criteria, we hypothesize that we will have a solid base for generating realistic fragments of music. We can insert snippets of the musical figures we want to detect into the fragments, in order to generate a complete dataset that

can be used for training. In contrast to the rules above, the figure we want to insert - the *figura corta* - is more rigorously defined, and will be implemented as described in section 2.1.2. The *figura corta* is a rhythmic figure, so only the melodic rules will be enforced, while the rhythmic rules are replaced by the description of the *figura corta*.

3.1.2 Generating data

In this section we will describe the methods we use to comply to the rules stated in the previous section, after which we will explain the algorithm that will procedurally generate the synthetic data. We will consider a fragment to be 2 measures of 4 quarter notes, only considering 4/4 time signatures. Subsequently, when testing on real data, we will also take 2 measures of 4 quarter notes at a time.

To ensure the first rule, we will consider three modes that were commonly used by Bach: The major mode, the minor mode and the melodic minor mode. Each of these scales can be described in terms of the chromatic intervals between the pitches, starting from the tonic. For example, the major mode will be characterized by the sequence $\{2, 2, 1, 2, 2, 2, 1\}$. Starting from the tonic, the next pitch will be two chromatic steps up, followed by two more steps, then one step, and so on. When applied to the tonic D, this will produce the scale $\{D, E, F\sharp, G, A, B, C\sharp, D\}$, or D major. We can continue this up and down the entire pitch domain to receive multiple octaves of the scale. Each fragment will draw pitches from this final set, satisfying our first rule.

The second rule we want to enforce is to generate melodies that prefer small intervals between the notes. In order to solve this, pitches will be chosen from a normal distribution centered on the index of the previous pitch, with available pitches equal to the set of pitches generated from the scale and tonal center of the fragment. We manually lower the chance of rolling the same pitch consecutively to avoid melodies staying on one pitch. By using a normal distribution, larger intervals are still possible (as they should be), but they get exceedingly rarer as the interval gets bigger. Most pitches will however lie close to the previous, allowing the melodies to roughly follow the outline of the scales, which often happens in Bach's cantatas.

For the third rule, we will implement three different approaches, in order to get a better understanding of what makes a rhythm realistic. We will call these three methods *random*, *divided* and *sampled*:

- **Random.** The first approach is based on the fact that quarter- and sixteenth notes seem to be the most common. We will use a Poisson distribution¹ to generate a number between 0 and 5 (both included), which correspond to sixteenth-, eighth-, quarter-, half-, whole- and two whole notes respectively. The Poisson distribution will have a λ of 1, which makes eighteenth- and eight notes equally probable at a probability of approximately 37% each. After that, the distribution quickly falls. Since the Poisson distribution can theoretically output any natural number, we will re-roll the distribution if the result exceeds two whole notes.
- **Divided.** The second approach involves recursively dividing each measure into parts. In every recursion, there is a 30% chance that it will just return the part as it is. Otherwise, it will split up into two parts, feeding each into the next recursion. The recursion stops when it receives a single sixteenth note. For an illustrated example, refer to figure 3.1.

¹A distribution defined as $P(X = k) = \frac{\lambda^k e^{-\lambda}}{k!}$ with parameter $\lambda > 0$

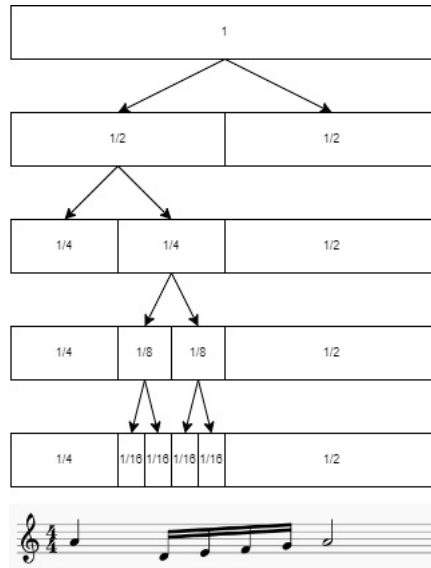


Figure 3.1: Recursively dividing a measure

- **Sampled.** The last approach is based on n-gram models, which are widely used in NLP (Sidorov, 2019). We will sample 2-grams from Bach’s cantatas, after which we compute the conditional probabilities $P(d_n|d_{n-1})$ as the amount of times duration d_n came after d_{n-1} divided by the total occurrences of d_{n-1} . Starting with a random duration (generated by the *random* method), we sample each next duration d_n from the conditional distribution over the previous duration d_{n-1} .

For the fourth rule, we decided upon a simple ratio between notes and rests. After manually assessing our generated data with 10%, 20% and 30% chance of a rest instead of a note, we found 10% to be too sparse and 30% to be too much. 20% seemed like a balanced middle ground, producing longer, uninterrupted melodies as well as short phrases divided by rests.

Finally, our fifth rule requires us to create fragments containing figures in 3 parts: starting with a random sequence using the methods above, then inserting the figure, and ending with another random sequence. At the start of generating a figure fragment, we first choose the length of the figure uniformly between 2 loops and 6 loops, with a maximum of 6 quarter notes. We define loop of a figura corta as a single short-short-long or long-short-short motif. After that, we choose the length of the first random sequence uniformly between 0 and the remaining length of the fragment. Lastly, we will fill the remaining part of the fragment with another random sequence.

In order to insert the figura corta, we use the chosen figure length and melodic scale. For the pitches, we use the same method as stated above, since the figura corta is purely a rhythmic figure. The basic premise to generating this figure is a small recursion: at any point, by looking at the previous two notes, we decide the duration of the current note: output a short duration if one of the previous two notes is a long duration, otherwise output a long duration. Repeat this until the figure is at the desired length. As a starting case, we randomly generate two durations, uniformly choosing between [short, short] and [short, long]. These do not contribute to the final output, but act as a starting point for the recursion. We define short durations as 16th notes and long durations as 8th notes. When going for a generalized figura corta, we choose a factor of 0.5, 1 or 2 beforehand and multiply the final output by that factor. This ensures we can get three variations:

one using 32nd and 16th durations, one using 16th and 8th notes and one using 8th and quarter notes. We use a poisson distribution to determine the factor, favouring factors 0.5 and 1, since those are the most common.

When combining all stochastic processes described in this section, we have proposed an answer to research question 1.2. See Algorithms 1 and 2 for an abstract overview of the generation process for random sequences and the figura corta respectively. Examples of a generated random (no-figure) fragment and a generated figura corta fragment are seen in figures 3.2 and 3.3.

Algorithm 1 Creating a random sequence

Input: *TotalDuration*, *Pitch*, *Scale*, *DurationMethod*
Result \leftarrow Empty List
RemainingDuration \leftarrow *TotalDuration*
while *RemainingDuration* $>$ 0 **do**
 Pitch \leftarrow NextPitch(*Pitch*, *Scale*)
 Choose *Duration* using *DurationMethod*
 Word \leftarrow Note(*Pitch*, *Duration*) or Rest(*Duration*) \triangleright 20% chance of a rest
 Add *Word* to *Result*
 RemainingDuration \leftarrow *RemainingDuration* $-$ *Duration*
end while
 Return *Result*, *Pitch*

Algorithm 2 Creating a figura corta

Input: *AmountOfLoops*, *Factor*, *Pitch*, *Scale*
Result \leftarrow Empty List
 Choose *BaseCase* uniformly from $\{[1/4, 1/2], [1/4, 1/4]\}$
PreviousDurations \leftarrow *BaseCase*
i \leftarrow 0
RemainingDuration \leftarrow Min(6, *AmountOfLoops* * *Factor*)
while *i* $<$ *AmountOfLoops* & *RemainingDuration* $>$ 0 **do**
 Pitch \leftarrow NextPitch(*Pitch*, *Scale*)
 if $1/2 \in$ *PreviousDurations* **then**
 Duration \leftarrow 1/4
 else
 Duration \leftarrow 1/2
 end if
 Remove oldest element from *PreviousDurations* and add *Duration*
 CurrentNote \leftarrow Word(*Pitch*, *Duration* * *Factor*)
 Add *CurrentNote* to *Result*
 i \leftarrow *i* + 1/3 \triangleright 3 notes form one loop
 RemainingDuration \leftarrow *RemainingDuration* $-$ *Duration* * *Factor*
end while
 Return *Result*, *Pitch*

3.1.3 Data representation

Since LSTM models are found to be successful in NLP tasks, it seems fit to transform the musical data in such a way that the task of detecting musical figures becomes



Figure 3.2: A fragment generated by our algorithm (sampled durations), containing no figura.

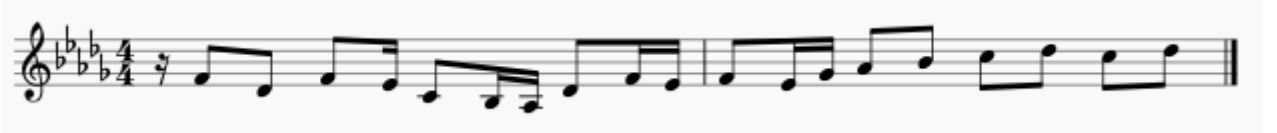


Figure 3.3: A fragment generated by our algorithm (sampled durations), containing the figura corta.

comparable to a text classification problem. To achieve this, first we define a musical equivalent of a word. The easiest way to think about a *word* in a musical context is to look at individual notes and rests, with rests acting as notes without a pitch. Multiple *words* played in sequence can form a musical *fragment* - which would be the equivalent of a *sentence* in NLP - and a sequence of multiple of these fragments form a musical *piece*, or *document* in NLP.

In NLP, word embeddings are used to represent a word as a vector in a space where words with similar meaning and syntax are close together (Ghannay et al., 2016). These are often pretrained on large corpuses and serve as means to present a higher dimensional input to a model. In our representation of symbolic music, individual words don't have an independent meaning, since its semantics are determined by the context surrounding it. Furthermore, these words can already be written as a higher dimensional vector, since a single note or rest has multiple properties. These can be combined in a vector to serve as input. For our purpose, we will use two properties to describe each word in a fragment:

1. Chromatic pitch \vec{p} : this is the absolute pitch of a note based on the 12-TET system². We can follow the MIDI standard, where the C4 is represented by the number 60 and every half step up or down adds or subtracts 1 respectively. However, this could result in the models treating higher notes as "more active" or "better", so we will instead represent the pitch as a one-hot vector. The MIDI standard contains a lot more pitches than are present in the cantatas, however, so we will treat 0 as the lowest note in our available data (which corresponds to a G1), and notes will have a maximum pitch of 62 (which corresponds to an A6). Rests have no pitch, so their pitch will be represented by the zero vector.
2. Duration $d \in \mathbb{Q}^+$: this value is also represented by a one-hot encoding, with the total length of this vector equal to the amount of unique durations we have seen in the cantatas.

This results in a vector $\begin{bmatrix} \vec{p} \\ d \end{bmatrix}$ of length 115 for each word.

²Twelve-Tone Equal Temperament. This is the system of dividing an octave into 12 pitches which was used in the entirety of Western European classical music, and still is.

3.2 Detecting the figura corta

Now that we have the ability to generate an extensive dataset, we can focus on using the LSTM architecture as a suitable classifier, answering research question 2.1. Afterwards, we will look at the different parameters we will vary in search of an optimal model, as an answer to research question 2.2.

3.2.1 LSTM architecture

As a baseline, we will input musical sentences $[x_1, \dots, x_N]$ as given by the data generation algorithm into the LSTM model, and the model will be treated as a classifier with C classes corresponding to $C - 1$ different figures and one class representing the absence of all figures. In our case, where we only treat the figura corta, this means that it becomes a binary classifier: a fragment either contains a figura corta or it doesn't. We will adapt the LSTM architecture to turn it into a suitable classifier.

Our LSTM model will take inputs of length 115, corresponding to the vectors produced by our representation. After that, each step in the sequence will produce a hidden state of 8 or 32 units, which will be discussed in the next section. After processing the entire input sequence, the LSTM will output its last hidden state. We use this output as the input for a last fully connected layer with softmax activation to receive our final output vector of size C corresponding to a probability distribution over all possible classes (in this case 2). We will take the index of the maximal value in this vector as our final prediction. For training, we use the Adam optimization algorithm (Kingma & Ba, 2014) with cross entropy loss.

3.2.2 Parameters

When generating data and training an LSTM, there are some choices involved. We are going to explore the influence of these parameters by training our LSTM on different datasets generated by different variations of the algorithm and assessing the accuracy of each model. Some choices are fixed in every variation of the algorithm (like the ratio between notes and rests), while others will be implicitly tested by the model. This is done to avoid exponentially increasing the number of tests to be performed. We will vary three parameters.

The first parameter is the way in which we generate durations: *random*, *divided* or *sampled*, each described in the previous section. We use the *random* method as a baseline to test the other two against, since we hypothesize that the actual durations do not follow a statistical distribution, but are rather influenced by their surrounding notes and place in the measure.

Secondly, we will train on datasets containing *simple* or *generalized* figura corta. In the *simple* variation, we always generate the figure using two sixteenth notes and one eighth note. Although very common, it is not the only way in which the figura corta is used. Thus, the *generalized* method of generating will randomly vary the figura corta durations as described in 3.1.2. This parameter will give us insight in the model's capability of generalization.

The last parameter we will tweak is the hidden state size of the LSTM. We will consider sizes 8 and 32. The difference in accuracy between these two models can give insight in the complexity of the data, while we can also assess the training speed between the two.

3.3 Evaluation

This section will answer our last research subquestion 3.1. First, we will test the different parameter settings on the synthetic data. In total, we have 12 different combinations of synthetic data and model structure. For each combination we will generate 10,000 fragments, with each fragment containing either a figura corta or no figure with equal probability. We then split the dataset into 80% training and 20% validation data. After training on the training data, we test the model on the remaining validation data to see how it performs on data it has never seen before. We will report the validation accuracy, as well as the average accuracy during training for each combination. The difference between the average accuracy during training and the final validation accuracy can tell us a lot about the potential of the model. As counter-intuitive as it may sound, we don't just want 100% accuracy during training and validation. Since we generate our own data, this is very easy to achieve by just making the data wildly different depending on the label. We postulate that finding the figura corta is not a trivial task, so we expect the model to not immediately get it right. When the average accuracy during training is close to the validation accuracy, it signifies that the model spent most of its training close to the validation accuracy, and that it does not possess much potential to further improve on its accuracy. We hypothesize that a model with a low average accuracy during training but a high validation accuracy sufficiently struggled with the task, making the synthetic data closer to the reality.

The difference in performance between size 8 and size 32 LSTM models is also a relevant metric for assessing the complexity of the data. When utilizing a bigger hidden layer size, the model can deduce more intricate connections between different parts of the data. If there is a significant improvement when using a 32 size LSTM compared to an 8 size LSTM, there are more potential connections to be learned. This correlates with the complexity of the data, and assuming the final accuracy is still satisfactory, we hypothesize that the model's learned features will be better transferable to the real data.

After evaluating the models on the synthetic data, we will take our best performing parameter combination and use these parameters to train a model on an extended synthetic dataset of 40,000 training fragments. The resulting model will be tasked with annotating multiple real cantata movements in search of the figura corta. Starting on the first measure, each set of two measures starting with an uneven one will constitute a fragment, i.e. measures {1 and 2}, {3 and 4} but not {4 and 5}. In order to extract meaningful results, we will also manually annotate these movements beforehand using the same fragments, highlighting the fragments where there is a visible figura corta present. Then, the model will perform the same task, highlighting the fragments where there is an output of at least 0.5 corresponding to a figura corta. The cantata movements we will be annotating are:

- BWV 12, movement 4
- BWV 21, movement 2
- BWV 31, movement 4
- BWV 63, movement 3
- BWV 63, movement 7
- BWV 132, movement 5

These particular movements are known to contain at least one occurrence of the figura corta, and one movement, BWV 63 mov 7, sporting a significant amount of occurrences

throughout the piece (Van Kranenburg, 2010). BWV 63 mov 7 will act as a larger stress test for the model.

For each cantata movement we evaluate, we will draw up a confusion matrix consisting of the true positives, false positives, true negatives and false negatives that the model outputs. From this confusion matrix, we deduce three different metrics: precision, recall and accuracy.

The precision, defined as the ratio of true positives to the total positives given by the model, will tell us what part of the returned figures are legitimately assigned. When the model is used as a tool for retrieving figures to analyze them further (i.e. manually), a high precision is valuable, since it reduces the need to constantly check if the returned fragment does indeed contain the requested figure.

The recall, which is the ratio of true positives to the total amount of figures in the movement, indicates how thoroughly the model searches for the figures. A high recall is extremely useful if the aim is to produce an exhaustive list of every figure present in a cantata. Furthermore, when performing further automated analysis on the figures, a high recall prevents that analysis from missing valuable data.

The last metric we will report is the accuracy. This is simply the ratio of the total amount of truthful statements by the model to the total amount of statements, be it true positives or true negatives. The accuracy will act as a sanity check, a quick way to notice when the model gives unusually poor results. The cantata movements we will annotate will contain lots of fragments and figures are usually sparse, which means that most fragments will not contain a figure. Considering this, the model can actually achieve a reasonable accuracy by never returning a figure. Therefore, the accuracy is a less relevant metric in this case, except for the reason described above.

Chapter 4

Results

In this chapter, we will first test each of the 12 different parameter settings as described in the last section. The accuracy of each trained model will be reported and interpreted, and the best performing model will be chosen to search for the figura corta in Bach's music.

4.1 Accuracy on synthetic test data

Table 4.1 shows the accuracy of all 12 parameter settings that were tested. One thing to note here is that the average accuracy while training shows the overall accuracy over all 8000 training fragments. This includes the beginning of the training, which will generally drag the total accuracy down. This explains why, in every setting, the validation accuracy is strictly better, because the model will assess all 2000 validation fragments with a trained network.

While the validation accuracy is the main metric that determines the performance of the model, the difference in training and validation accuracy tells us a lot about the complexity of the data and the potential of the model. When the validation accuracy is close to the average training accuracy, we can deduce that the model spent most of its training time close to its actual performance, meaning that it converged rather quickly. This implies that the model does not have much potential to improve, even when exposed to more training data. Secondly, a quick convergence in combination with a high accuracy

Duration method	Figura Corta	LSTM layer size	Training average	Validation
Random	Simple	8	0.8643	0.8955
Random	Simple	32	0.8798	0.914
Random	Generalized	8	0.8388	0.9130
Random	Generalized	32	0.8586	0.9170
Divided	Simple	8	0.8844	0.9120
Divided	Simple	32	0.9019	0.9530
Divided	Generalized	8	0.9196	0.9320
Divided	Generalized	32	0.9196	0.9360
Sampled	Simple	8	0.8259	0.9220
Sampled	Simple	32	0.8826	0.9555
Sampled	Generalized	8	0.7525	0.8945
Sampled	Generalized	32	0.8154	0.9435

Table 4.1: Accuracy of the 12 different parameter settings



Figure 4.1: A misidentified pattern in BWV 63 mov 3

also tells us that the data on which the model trained might have been too straightforward. This can be seen in the parameter settings where we chose the durations randomly or divided. In five out of eight cases, there was less than 5% improvement between the average training and validation data, while most validation accuracies lie above 90%. However, in the case of *sampled* durations, all settings achieve at least an 8% improvement between average training and validation accuracy, with two of the cases achieving at least 15% improvement.

4.2 Results on real data

For testing on real data, we will train a new model on 32,000 fragments. We will train a size 32 LSTM on datapoints generated with *sampled* durations and generalized figura corta, based on this combination’s high accuracy and its ability to generalize the figura corta well. This final model achieved a validation accuracy of 0.9682 on 8,000 synthetic data points, and was further tested on the six cantata movements as described in section 3.3. The resulting confusion matrices are reported in this section, as well as a combined confusion matrix containing the sum of all the results. The overall accuracy of the model when tested on the cantatas is 94.46%, which is really close to the 96.82% accuracy of the model when validated on the synthetic dataset. However, as stated before, the accuracy mostly acts as a sanity check in order to quickly detect anomalies.

4.2.1 Precision

Looking at the combined confusion matrix (figure 4.9), the average precision over all tested movements is 78.53%. Although this implies that we can expect nearly 80% of the positive output to be correct, we do see that the precision heavily differs between movements. The real outlier is BWV 63 mov 3 (figure 4.6), in which the model only achieved 52.38% precision, meaning that half of the predicted figures were not assigned that label upon manual annotation. After further inspection, half of the false positives were due to a single pattern that the model misidentified as a figura corta, namely a sequence where a quarter note was followed by an eighth rest and an eighth note (see figure 4.1). As for the rhythmical part of this sequence, it follows the pattern of the figura corta. However, the actual perceived rhythm in this sequence differs greatly from the figura corta, since the rests in between the notes alter the rhythmic structure.

4.2.2 Recall

The model achieved an average recall of 83.12%, meaning that only a little more than 15% of the actual figures were not detected. In four out of six movements, the model actually returned all of the figura corta in the movement, that being BWV 12 mov 4 (figure 4.3), BWV 31 mov 4 (figure 4.5), BWV 63 mov 3 (figure 4.6) and BWV 132 mov 5 (figure 4.8). Again, there is only one real outlier, which is BWV 21 mov 2 (figure 4.4). While it boasts a high precision of 94.44%, its recall is only 68%. The figures it missed look



Figure 4.2: An identified and a missed figure in BWV 21 mov 2

		True class	
		1	0
Predicted class	1	21	5
	0	0	31

Precision = 80.77%
 Recall = 100%
 Accuracy = 91.23%

Figure 4.3: Results of annotating BWV 0012 mov 04

rather arbitrary, especially given the fact several of those figures are simply the same as other figures that the model did find, except shifted in pitch. The real difference between hit or miss seems to be regarding the context surrounding the figures. In two particular cases, there is a dotted quarter note after the figure, which might throw the model off. One of these cases can be seen in figure 4.2, where the top figure has been identified correctly, but the bottom figure has not. The figures themselves follow the same melodic trend, but the detected figure is 1.5 times the length of the missed figure. Do mind that the model is capable of detecting figures where there are only two repetitions of the basic figure, as seen in the other annotated movements.

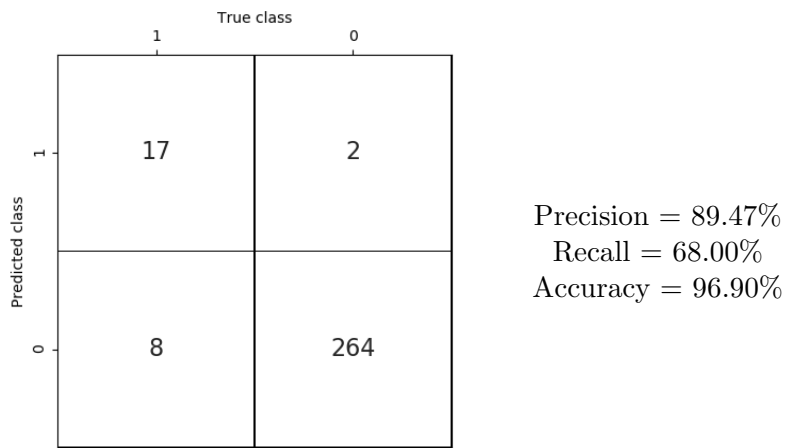


Figure 4.4: Results of annotating BWV 0021 mov 02

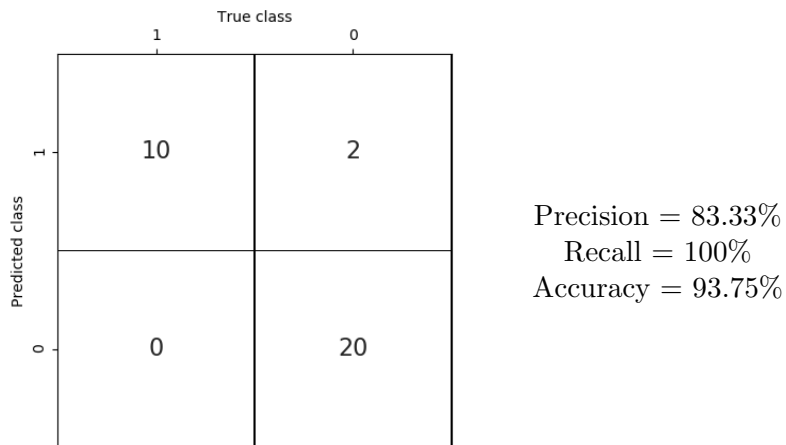


Figure 4.5: Results of annotating BWV 0031 mov 04

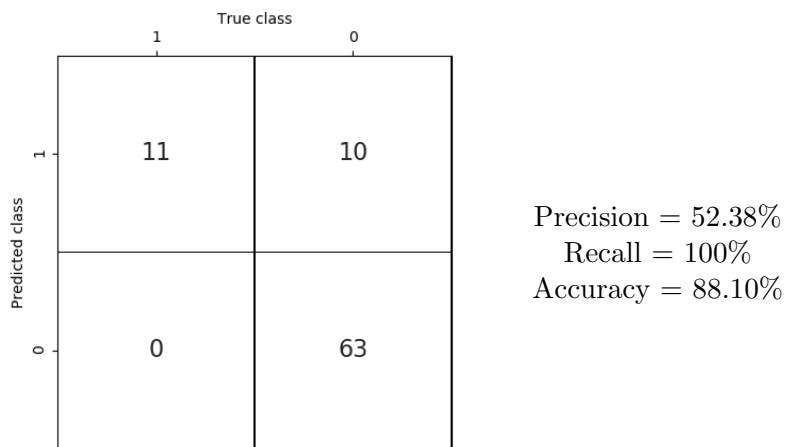


Figure 4.6: Results of annotating BWV 0063 mov 03

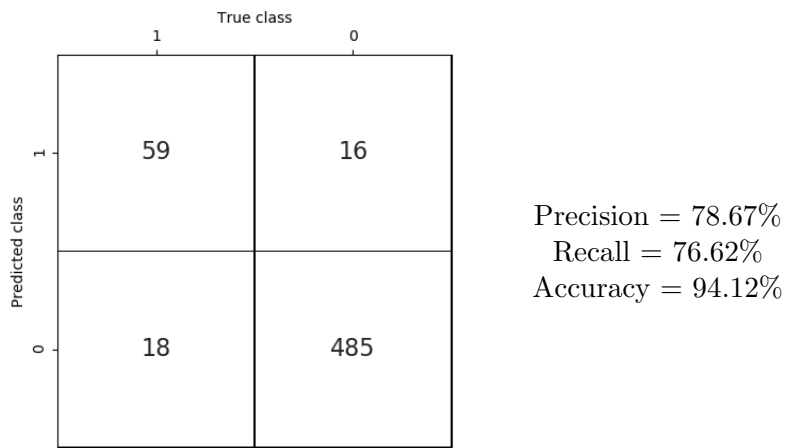


Figure 4.7: Results of annotating BWV 0063 mov 07

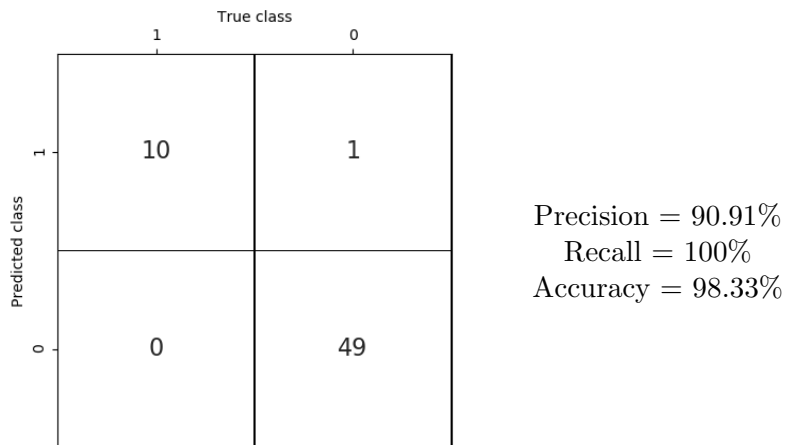


Figure 4.8: Results of annotating BWV 0132 mov 05

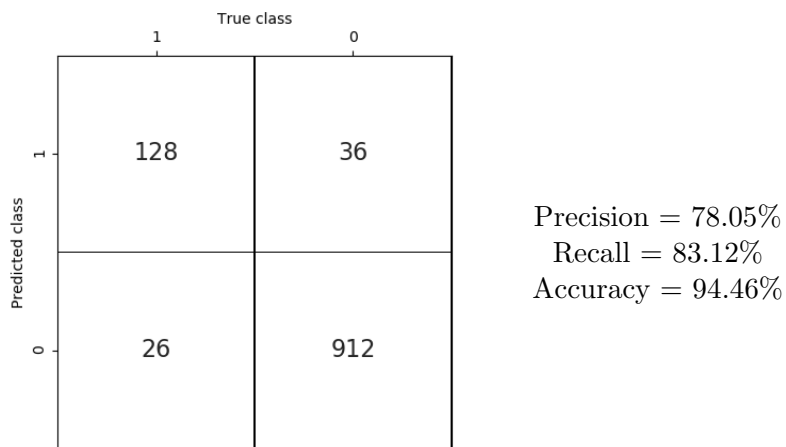


Figure 4.9: Results of all annotated movements combined

Chapter 5

Conclusion

In this thesis, we researched whether it is possible to detect the figura corta in Bach's cantatas, using only synthetic data as a training source. By drawing up laws and rules to which the data should abide, we managed to create an algorithm that generates musical fragments. When manually inspecting the generated figures, we see plausible melodic lines with realistic rhythms, which in most cases could very well appear in a real cantata. We then explored twelve different parameter settings by training an LSTM on the resulting datasets. We performed a simple grid search, comparing three different ways of generating note durations, two ways of generating the figura corta and two hidden layer sizes for the LSTM. After considering validation accuracy and the models' ability to learn, we chose the most optimal combination of parameters. Note durations were generated dependent on the duration of the note before, the figura corta could occur in three ways, each with a different duration factor, and we chose an LSTM with a hidden layer size of 32. We trained an LSTM on an extended dataset generated by our algorithm to receive our final model. This model was tasked with annotating six cantata movements by Bach¹ in search of the figura corta. These movements are comprised of 1101 fragments of two measures containing a total of 154 figura corta. We manually annotated these same cantata movements in order to compare the two annotations. From this comparison, we drew up confusion matrices and calculated the precision, recall and accuracy for each movement.

Taking a weighted average over all six cantata movements, the model achieved a precision of 78.53%, a recall of 83.12% and an accuracy of 94.46%. Looking at the movements individually, we see several cases where the model did not miss any of the figura corta present, giving a recall of 100%. In our introduction, we proposed the following research question: *Is it possible to reliably detect the figura corta in Bach's cantatas with an LSTM by training the model on a synthetic dataset?* From the results above, we can conclude that synthetic data *can* produce a reliable dataset to train an LSTM that can successfully classify real data on whether it contains a figura corta or not. Furthermore, this result implies that our method of procedurally generating data produces a varied dataset that can train a model to be capable of generalization. By extension, it implies that the laws and rules we proposed, as well as the representation of the musical data, allow the LSTM to correctly apply its learned features to real data.

¹BWV 12 mov 4, BWV 21 mov 2, BWV 31 mov 4, BWV 63 mov 3, BWV 63 mov 7 and BWV 132 mov 5

Chapter 6

Discussion

In the previous chapter, we concluded that our method of generating synthetic data can produce viable training data for an LSTM, which can then quite reliably detect the *figura corta* in Bach's cantatas. In this chapter, we will discuss some critical notes regarding our methods, as well as take a closer look at the mistakes that the model made. We have conducted an extra experiment that will evaluate how Bach-like our synthetic data actually is. The methods, results and conclusion for this small experiment will be described in section 6.1. Afterwards, in section 6.2 we will analyse the shortcomings of the model that came forward in the results and propose possible changes to combat those shortcomings. Lastly, we will complete this thesis by providing a perspective for future research in section 6.3.

6.1 Realism of synthetic data

Most of the criteria laid out in section 3.1.1 were meant to enhance the plausibility of the generated fragments. This was done in order to create a dataset that closely resembled the unlabeled real cantatas, so that training on this dataset would be comparable to training on the actual cantatas. Since the tests performed on the cantatas produced desirable results, we concluded that this dataset was indeed sufficiently real, as the model did not noticeably get confused by the real fragments. However, this does not actually imply that the synthetic data is indistinguishable from Bach.

In order to quantify how realistic the synthetic data is, we can train a new LSTM model on a combination of synthetic and real data, with the goal of classifying the input as synthetic or real. This method is inspired by the Generative Adversarial Network (GAN) framework (Goodfellow et al., 2014), in which two machine learning models compete with each other: one model generates data (the generative model), while the other attempts to classify whether its input is real or generated (the discriminative model). In the ideal case, the discriminative model will not be able to distinguish between the generated and real data, meaning that the generated data looks just as real as the real data. In that case, the discriminative model will have an accuracy of 50%.

As a small experiment, we have trained a discriminative LSTM model to see whether it can find significant differences between the synthetic data we generated in chapter 3, and fragments from the actual cantatas. We used a training dataset that consists of 50% real data and 50% synthetic data. The real part of the data consists of fragments of 2 measures, and was randomly sampled from the set of cantatas. About 20% of the available music was used, as to avoid the dataset becoming unnecessarily large. Since cantata movements can drastically differ in length, it would be fair to sample more fragments from the larger

movements, as they contribute more to the total length of all cantatas. To achieve this, we sampled a different number of fragments from each movement, equal to the total movement length in measures divided by 5.

The synthetic data was generated without any figures deliberately inserted, to avoid overfitting on those specific figures (which can be detected in the synthetic data, as concluded in this thesis). We generated exactly the same amount of 2-measure fragments as sampled from the cantatas to produce a fully balanced dataset.

In total, sampling roughly 20% of the cantatas and adding an equal amount of randomly generated fragments, we ended up with 18,006 fragments. The entire dataset was then split into 80% training and 20% test data. After training an LSTM model with hidden layer size 32 on the training data, the resulting model achieved an accuracy of 0.8648.

This means that in more than 85% of the fragments, the model could reliably predict whether the fragment was real or fake. Using this statistic, we can safely conclude that our synthetic data is in fact not very comparable to Bach. This begs the question whether it actually matters how Bach-like the synthetic data is. Of course, in an ideal situation, we would have a completely labeled set of real cantatas to train on. That would almost certainly produce a very accurate LSTM model, given the effectiveness of LSTM as a neural network architecture. To contrast this, a synthetic dataset where every note is totally random in pitch and rhythm - except for the ones that constitute a figure - would probably result in a very inaccurate model when tested on Bach. This is due to the fact that Bach's music would be vastly different from any of the training fragments, for both of the labels. From the results presented in this thesis, it seems that the criteria - and subsequently the algorithm - produce a dataset that is realistic enough for a trained model to transfer the learned features to Bach, but not realistic enough to fool a simple LSTM. While more Bach-like synthetic data could lead to an improvement to the accuracy of the model when tested on Bach, it could also lead to a less generalized model. The model that was trained in this thesis could be applied to other styles of baroque music or even western classical music from other periods in time.

6.2 Error analysis and improvements

In this section, we will discuss some improvements that could be made to the method presented in this thesis. While the overall results were satisfactory, there were some outliers and cases where the model consistently failed to correctly identify the figura corta. One of these cases was highlighted in figure 4.1, where the model wrongly assigned the figura corta label. When listening to this fragment, it sounds obvious as to why this is not a figura corta. The rests in between the notes alter the rhythmic feel of the fragment, where there are only two notes heard instead of the three notes that constitute the figura corta. However, it is clear how the model got to its conclusion: when purely looking at the duration of the figure, we see four loops, each consisting of one quarter length followed by two eighth lengths. While the model understood the rhythmic pattern, it failed to learn that the figura corta does not contain rests, even though the synthetic data only presented the figure without rests. One way of solving this problem is to insert counterexamples in the synthetic data, based on this misidentified fragment. However, that would add to the manual labour required to craft the synthetic data, especially when introducing counterexamples to every false positive that is discovered when testing on more cantatas. When looking for the cause of the problem, we hypothesize that the model simply ignores all pitch-related data when assessing the figura corta. While this way of

thinking is right on a surface level, since the figura corta is a purely rhythmical figure, the one true edge case is when there is *no* pitch. This problem might actually solve itself when introducing other figures than the figura corta to the equation. When the model must focus on pitch-related data for other figures, it may also deduce that the figura corta should have a pitch for every note in the figure.

The second outlier when testing the model was in BWV 21 movement 2 (see figure 4.2). In this cantata, the model missed more than 30% of all figures. The interesting part about this low recall is that the model did manage to find the figura corta in fragments that are very similar, as seen in the figure. We believe that this is due to the fact that the length of the fragments is too long in some cases. The two figura corta both occur at the end of a two-measure fragment, meaning that there have been many distracting notes and rests beforehand. The fact that the upper figure contains three loops, in contrast to the two loops below, may have solidified the model's belief that it is in fact a figura corta. The two loops of the missed figure could have been insufficient support for the figure considering everything it has already seen in the fragment. We originally chose to consider fragments of two measures for the simple reasons that the figura corta could span more than one measure and it could even occur at the boundary of two measures. When one loop of the figure falls just before the end of a measure and the second loop starts in the next measure, the model can only pick up the figure when considering both measures. A different approach to this could be to vary the length of the fragments in the training data, and allowing the model to analyze the cantatas using a sliding window instead of two measures at a time. While increasing the time to annotate a cantata movement (since it will consider every note multiple times), it could also result in more precise annotation.

6.3 Future work

6.3.1 Using GAN for synthetic data

In section 6.1, we tested for realism in our synthetic data using a discriminative classifier inspired by the GAN architecture. Continuing this line of thinking, we can actually use the other half of the GAN principle as well when generating data. When pitting a generative and a discriminative model against each other with their tasks being generating realistic data and distinguishing fake from real data respectively, the generative model could actually provide us with realistic, Bach-like data. GANs have already been used for both data augmentation (Bowles et al., 2018; Frid-Adar et al., 2018; Kukreja et al., 2020) and generating synthetic data (Bhattarai et al., 2020; Jordon et al., 2018), but mostly in the field of Computer Vision. We could task the generative model with generating random sequences of different lengths that conform to the laws stated in section 3.1.1, while the discriminative model can compare those to fragments from the real cantatas. If the discriminative model converges to 50% accuracy, the generative model should be able to produce realistic, random fragments that can be used instead of the algorithm we designed in section 3.1.

6.3.2 Detecting other figures

The figura corta was chosen as the object of this thesis for several reasons. First, it can be clearly described in terms of a pattern, which makes generating it fairly straightforward. Secondly, it has multiple forms it can take on, beginning with either two short durations or a long duration, and also varying between different note durations altogether. This allowed us to test the LSTM's capability of generalization when training on the synthetic

data. Lastly, the *figura corta* has plenty of documented occurrences in Bach's work, which made it a great candidate for testing against a ground truth. For a first exploratory work, it demonstrated the potential of procedurally generated synthetic data in combination with an LSTM, while not being overly complex in its execution. However, the *figura corta* is certainly not the only figure that can be detected using this method. One such figure is the *anabasis*, described by Walther (1732) as a rising motion, in melody represented as a rising pitch. Its counterpart, the *katabasis* is a downwards melodic line, which would pose the same complexity for the LSTM. Further figures include the *circulatio*, a figure that describes a circling motion (Dings, 2022). The *circulatio* is a melodic figure centered around one pitch, where the melody can circle that pitch in multiple ways: by going up in pitch, then down and eventually up again, or vice versa (when read from left to right, the figure resembles a wave motion). Each of these figures can reuse most of the synthetic data generation that we used in this thesis, but the rules for the figures themselves will have to be implemented once more, including stochastic variation. The fragments containing these figures can become part of the same dataset, as it will just add more classes to the labels, and thus to the LSTM classifier. One challenge in adding more figures is the fact that the figures might overlap. For example, an *anabasis*, a melodic figure, can easily coincide with the *figura corta*, since the definition of the *figura corta* does not consider any melodic movement. This means that one fragment might contain multiple figures, be it accidentally or on purpose. This might require altering the LSTM architecture to allow for multiple classes per fragment, and/or altering the synthetic data generator to ensure that it does not accidentally insert figures into fragments that do not have that particular label.

6.3.3 Analysis of musical rhetoric

Wrapping up, we will reflect upon the motivation of our research: the study of the relation between rhetoric and music in the baroque period. When trying to find a particular meaning behind a musical rhetoric figure, one could look at text that is sung with the figure either in the melody or the accompaniment. Not unlike manually detecting the figure, analysing the corresponding text by hand is a time-consuming task. However, with the possibility to automatically annotate cantata movements, automatic analysis of the text is not that far away. When our model is better capable of pinpointing the *figura corta* in a smaller time interval - for example by using a sliding window when annotating - we could take the words that are being sung during the *figura corta* and put them together in one big database. This collection of *figura corta* related texts can be analysed by use of NLP techniques. One example would be to apply a topic model like Latent Dirichlet Allocation (LDA) (Blei et al., 2003) to find common themes. Given a large enough corpus of text belonging to the *figura corta* - obtained by applying the method in this thesis to annotate all cantatas - and enough support for certain themes occurring in that corpus, one could empirically prove a connection between the figure and its meaning.

References

- Bartel, D. (1997). *Musica poetica: Musical-rhetorical figures in german baroque music*. U of Nebraska Press.
- Benitez Jr, V. P. (1985). *Musical-rhetorical figures in the" orgelbuchlein" of js bach*. Arizona State University.
- Bhattacharai, B., Baek, S., Bodur, R., & Kim, T.-K. (2020). Sampling strategies for gan synthetic data. *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2303–2307.
- Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *the Journal of machine Learning research*, 3, 993–1022.
- Bowles, C., Chen, L., Guerrero, R., Bentley, P., Gunn, R., Hammers, A., Dickie, D. A., Hernández, M. V., Wardlaw, J., & Rueckert, D. (2018). Gan augmentation: Augmenting training data using generative adversarial networks. *arXiv preprint arXiv:1810.10863*.
- Choi, K.-Y., Coüasnon, B., Ricquebourg, Y., & Zanibbi, R. (2018). Music symbol detection with faster r-cnn using synthetic annotations. *1st International Workshop on Reading Music Systems*.
- Choi, K.-Y., Coüasnon, B., Ricquebourg, Y., & Zanibbi, R. (2019). Cnn-based accidental detection in dense printed piano scores. *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 473–480.
- Cifka, O., Şimşekli, U., & Richard, G. (2019). Supervised symbolic music style translation using synthetic data. *arXiv preprint arXiv:1907.02265*.
- Couch, L. W. (1984). Musical rhetoric in three preludia of dietrich buxtehude. *The Diapason*, 91(3), 14–15.
- De Campos, T. E., Babu, B. R., Varma, M., et al. (2009). Character recognition in natural images. *VISAPP (2)*, 7(2).
- Dings, M. (2022). Kleines lexikon der musikalisch-rhetorischen figuren. *Hochschule für*.
- Downie, J. S. (2003). Music information retrieval. *Annual review of information science and technology*, 37(1), 295–340.
- Finkensiep, C., Déguernel, K., Neuwirth, M., & Rohrmeier, M. (2020). Voice-leading schema recognition using rhythm and pitch features. *International Society for Music Information Retrieval Conference*.
- Frid-Adar, M., Klang, E., Amitai, M., Goldberger, J., & Greenspan, H. (2018). Synthetic data augmentation using gan for improved liver lesion classification. *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, 289–293.
- Futrelle, J., & Downie, J. S. (2002). Interdisciplinary communities and research issues in music information retrieval. *ISMIR*, 2, 215–221.
- Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to forget: Continual prediction with lstm. *Neural computation*, 12(10), 2451–2471.

-
- Ghannay, S., Favre, B., Esteve, Y., & Camelin, N. (2016). Word embedding evaluation and combination. *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, 300–305.
- Giraud, M., Groult, R., & Levé, F. (2012). Subject and counter-subject detection for analysis of the well-tempered clavier fugues. *International Symposium on Computer Music Modeling and Retrieval*, 422–438.
- Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2014). Generative adversarial nets. *Advances in neural information processing systems*, 27.
- Gupta, A., Vedaldi, A., & Zisserman, A. (2016). Synthetic data for text localisation in natural images. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2315–2324.
- Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8), 1735–1780.
- Janssen, B., De Haas, W. B., Volk, A., & Van Kranenburg, P. (2013). Discovering repeated patterns in music: State of knowledge, challenges, perspectives. *Proc. of the 10th International Symposium on Computer Music Multidisciplinary Research, Marseille, France, 20*, 74.
- Jordon, J., Yoon, J., & Van Der Schaar, M. (2018). Pate-gan: Generating synthetic data with differential privacy guarantees. *International conference on learning representations*.
- Katsiavalos, A., Collins, T., & Battey, B. (2019). An initial computational model for musical schemata theory. *ISMIR*, 166–172.
- Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Kiparsky, P., & Youmans, G. (2014). *Rhythm and meter: Phonetics and phonology, vol. 1* (Vol. 1). Academic Press.
- Kukreja, V., Kumar, D., Kaur, A., et al. (2020). Gan-based synthetic data augmentation for increased cnn performance in vehicle number plate recognition. *2020 4th international conference on electronics, communication and aerospace technology (ICECA)*, 1190–1195.
- Lartillot, O. (2005). Efficient extraction of closed motivic patterns in multi-dimensional symbolic representations of music. *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, 229–235.
- Lartillot, O. (2014). In-depth motivic analysis based on multiparametric closed pattern and cyclic sequence mining. *International Symposium on Music Information Retrieval: ISMIR*.
- Medsker, L. R., & Jain, L. (2001). Recurrent neural networks.
- Meyer, L. B. (1996). *Style and music: Theory, history, and ideology*. University of Chicago Press.
- Nikolenko, S. I. (2021). *Synthetic data for deep learning* (Vol. 174). Springer.
- Qi, S., Zhu, Y., Huang, S., Jiang, C., & Zhu, S.-C. (2018). Human-centric indoor scene synthesis using stochastic grammar. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5899–5908.
- Rodriguez, P., Wiles, J., & Elman, J. L. (1999). A recurrent neural network that learns to count. *Connection Science*, 11(1), 5–40.
- Rolland, P.-Y. (1999). Discovering patterns in musical sequences. *Journal of New Music Research*, 28(4), 334–350.
-

- Schedl, M., Gómez, E., Urbano, J., et al. (2014). Music information retrieval: Recent developments and applications. *Foundations and Trends® in Information Retrieval*, 8(2-3), 127–261.
- Schmieder, W. (1950). Thematisch-systematisches verzeichnis der musikalischen werke von johann sebastian bach: Bach-werke-verzeichnis;(bwv).
- Schweitzer, A. (1908). *Johann sebastian bach*. Breitkopf & Härtel.
- Shin, R., Kant, N., Gupta, K., Bender, C., Trabucco, B., Singh, R., & Song, D. (2019). Synthetic datasets for neural program synthesis. *arXiv preprint arXiv:1912.12345*.
- Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of big data*, 6(1), 1–48.
- Sidorov, G. (2019). *Syntactic n-grams in computational linguistics*. Springer.
- Symons, J. (2017). *A cognitively inspired method for the statistical analysis of eighteenth-century music, as applied in two corpus studies* (Doctoral dissertation). Northwestern University.
- Van Dijk, P., Van der Leeuw, G., & Leussink, J. (1981). *Musiceren als brugman*. KRO.
- Van Kranenburg, P. (2010). Mit fried'und freud'ich fahr dahin (bwv 616).
- Walther, J. G. (1732). Musikalisches lexikon. *Reprint 1953, Kassel, Germany: Barenreiter-Verlag*.
- Wang, Q., Gao, J., Lin, W., & Yuan, Y. (2019). Learning from synthetic data for crowd counting in the wild. *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 8198–8207.
- Wessel, F. T. (1955). *The affektenlehre in the eighteenth century*. Indiana University.
- Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A review of recurrent neural networks: Lstm cells and network architectures. *Neural computation*, 31(7), 1235–1270.
- Zaremba, W., & Sutskever, I. (2014). Learning to execute. *arXiv preprint arXiv:1410.4615*.
- Zhou, C., Sun, C., Liu, Z., & Lau, F. (2015). A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.