An exploration of the current state-of-the-art in automatic music transcription - with proposed improvements using machine learning

October 17, 2018
Lund University

Abstract

The research field of automatic music transcription has vastly grown during the 21st century, where the goal is to transcribe a polyphonic music signal into annotated sheet music. Within this field, the subproblem of fundamental frequency estimation in a piece of music is a difficult problem, e.g., due to dissimilar structures in signals from different instruments playing the same note. This becomes further convoluted in a polyphonic signal consisting of several notes, where the harmonic overtones of the notes interact. To solve this and other issues, machine learning techniques have furthered the research in music transcription, which is the main focus of this thesis. This is undertaken by comparing the best performing fundamental frequency estimators from recent years, mainly from MIREX competitions from 2015-2017. These are recreated and evaluated on a customized test set consisting of MIDI files of various instruments. The evaluation consists both of typical music transcription measures such as precision, recall and accuracy, but also by deeper analysis in order to find the large-scale structural biases. The evaluation of the tests herein shows that the best performing models are THK1 and CT1 from MIREX 2017 which are based on CNN. This work has identified some structural errors in these methods pointing out potential for further improvements. In addition, a novel approach of applying complex-valued neural networks in music transcription is also examined, by modifying research in an existing deep complex neural network model. The proposed and improved model finishes on third place in the evaluation, indicating that complex neural networks may develop the research area of music transcription even further.

A special thanks to my supervisor, Dr. Ted Kronvall for all support and a commitment far beyond what could be expected in finding solutions and improvements of this work. I also want to thank the system manager James Hakim for helping me out with daily computer related questions. Finally all researchers who kindly contributed with their models, receives my gratitude. Without any of you this thesis would not be the same.

Contents

1 Popular scientific summary (in Swedish)

Forskning kring att utvinna information om musik direkt från en ljudinspelning har pågått sedan slutet av 1900-talet. Många nya upptäckter har gjorts de allra senaste åren, dels på grund av att forskningsområdet har varit relativt outforskat men också då den tekniska utvecklingen har gått framåt väldigt kraftigt. Denna avhandling utforskar en gren inom forskningsområdet, kallad multipel fundamentalfrekvensestimation, där målet är att få en dator att omvandla musik i form av en ljudinspelning av flera samtidigt spelande instrument, till ett notblad med korrekt transkriberade noter. En fundamentalfrekvens motsvarar ljudvågornas svängningshastighet för varje ton i notsystemet. Rent intuitivt kan detta låta som magi, och som man kan förvänta sig finns det heller inte någon enkel formel som felfritt löser problemet. De lösningsmetoder som finns är snarare inriktade på att göra en så bra estimation av det korrekta notbladet som möjligt. Tillvägagångssätten består generellt sett av avancerade modeller som i flera steg omvandlar data och optimerar parametrar utifrån olika tekniska och musikteoretiska aspekter. Hur forskningens senaste modeller fungerar och hur bra de presterar är två ämnen som denna avhandling utreder.

Tillämpningsmöjligheterna för forskningsområdet är stora, givet att en modell skulle fungera i princip felfritt. En musiker skulle t.ex. kunna få ut noterna till vilken ljudinspelning som helst, vilket skulle vara uppskattat och praktiskt för musiker på alla nivåer.

Avhandlingen går inledningsvis igenom de tekniska och musikteoretiska delarna som de senaste forskningsmodellerna baseras på, samt vilka hinder man behöver få bukt med. En stor del av svårigheten i multipel fundamentalfrekvens-estimation ligger i att ljudet från flera olika instrument går in i varandra och frekvenser kan på så vis både förstärkas eller ta ut varandra. Vidare vet datorn inte vilken uppsättning av instrument som ljudet kommer från, samt hur många toner som spelas samtidigt. Ett nyligen applicerat segment inom forskningsområdet är machine learning genom så kallade neurala nätverk. Ett neuralt nätverk lärs upp till att automatiskt identifiera mönster i ett okänt dataset genom att anpassa parametrar på en stor mängd träningsdata. Hur neurala nätverk fungerar rent praktiskt och hur de appliceras för att estimera fundamentalfrekvenser är något som också utreds i denna avhandling.

Ett avsnitt är tillägnat åt att implementera en egen modell som baserar sig på insikter från existerande modeller. Den presenterade modellen använder sig av en variant av neurala nätverk som tillåter komplexa värden, det vill säga roten ur negativa tal. Detta är en nyintroducerad variant av neurala nätverk inom forskningsvärlden som inom bland annat bildigenkänning har visat bättre resultat än likvärdiga realvärda neurala nätverk. I avhandlingen görs en omfattande utvärdering där totalt 24 modeller testas på 12 musikstycken i olika stilar och instrumentkonstellationer. Den föreslagna modellen presterar väldigt väl och är enligt testerna den tredje bästa modellen.

Testerna används dels för att avgöra vilka modeller som presterar bäst, vilket ger ett mått på vad dagens forskning kan åstadkomma. Testerna används också för att analysera vilka frekvenser modellerna klarar av att estimera samt om det går att hitta några strukturella fel som de inte klarar av. Att grundligt utvärdera metoderna inom dagens forskning kan förhoppningsvis hjälpa till att vägleda framtidens forskning till att fortsätta göra nya framsteg.

2 Introduction

To automate music transcription would in many applications be appreciated and developing. Take for example a musician sitting next to the guitar writing a song. The melody which was just invented on the guitar turned out to be perfect and the musician starts to write down the notes as good as it was remembered. Imagine the musician using an automatic music transcribing device, the melody would instead simply appear on a music sheet just as it was played. Another example is a musician listening to a song which the musician wants to play himself. Instead of listening to the song recording in slow motion in order to write down the tones one by one, an automatic music transcriber would give the sheet music directly. The musician could instead spend the time on practicing the song directly.

So why don't everyone use an automatic music transcription device? The simple answer is that it doesn't exist, at least not a completely working one, at least not right now. The follow-up question is of course, when will it exist? This question can of course not be answered by a master student in mathematical statistics. What a master student can answer, which this thesis will be about, is how well the currently best music transcription devices perform statistically, how they work and perhaps how far from perfect they currently are. To determine the directions of further research, this is an important question to answer. This question is actually answered every year in a conference called MIREX where music transcription models are evaluated. But the MIREX evaluation is not optimal in order to lead the way of future research, which will be motivated herein.

Every year researchers send their novel algorithmic solutions to MIREX and the best performing one is decided. The issue is that the question "why" it performs the best, is not further analyzed. In this thesis a broader selection of models, more corresponding to what currently is available in the music transcription area, will be evaluated. Also, the results will be presented such that different features in the models may be analyzed. In order to limit this thesis, both models in MIREX which were participating 2014 and before as well as other models which haven't participated in MIREX, have been omitted. It should be mentioned that even if more models are evaluated in this thesis than in one single MIREX conference, there are published models claiming to perform state-of-the-art which are not considered in this thesis.

The most recent music transcription models are based on machine learning, which can be described as methods where a large amount of data is used to train a network consisting of often several thousands of parameters. Machine learning has previously been proved to be successful for image analysis, but recently also for music transcription. This thesis will study the novel approach of applying complex-valued neural networks to music transcription. Complex-valued neural networks is a newly discovered area which has been showed to enrich other applications such as image recognition [1]. This thesis will aim to answer if complex-valued neural networks may enrich the area of music transcription as well.

Comparable studies as this thesis has been made, for example [2] where many types of music transcription techniques are explained and compared, or [3] where the challenges for the automatic music transcription area are described and what needs to be developed in order to make further progress.

This thesis is aimed for two types of readers. The first is the dedicated researcher in the music transcription subject, for whom the test results might be the most interesting. The second is the student who is new to the area and wants to get a thorough review of the current research and the underlying theory. The thesis will be structured as follows: In chapter 3 the fundamental tools for music transcription will be described. Chapter 4 will describe what a music transcription model is supposed to perform and how it is evaluated. In chapter 5 all the considered music transcription models are presented. In chapter 6 statistical testing of the models is described. In chapter 7 the test results are presented, and finally in chapter 8 conclusions of the results are made.

3 Theory

In this section the basic theory behind music translation will be introduced: First a general review of what automatic music transcription is presented in Section 3.1. The steps are then further explained by analyzing the music signal using different methodologies, such as spectral analysis in Section 3.1.1, musical theory in Section 3.1.2, signal processing in Section 3.1.3, and technical difficulties that may appear in Section 3.1.4. Then the theory behind the pitch estimation is explained, which essentially amounts to different solutions to a signal processing classification problem. An approach called subharmonic summation is described in Section 3.2.1, whereafter an introduction to neural network in Section 3.2.2, followed by different types of neural networks in Section 3.2.3 and 3.2.4. The section ends with the translation of pitch probabilities to a music sheet in Section 3.2.5.

3.1 Introduction of Automatic Music Translation

Transcribing music is complicated, therefore the solutions may be very advanced. The solutions also differ depending on who tries to solve the problem. There are a couple of steps on the path to a complete music transcription model which the current research seems to have agreed on [2]. The researchers' attempts to optimize these steps will be explained in this section.

First of all, if one wants to understand music transcription, one needs to understand some theory about audio. The formal definition of audio, stated in [4], is vibrations which propagate through a transmission medium such as a gas, liquid or solid. As further explained, humans can only hear these vibrations if the waves cycles in the specific range 20 Hz to $20 \cdot 10^6$ Hz (Hz = cycles per second). Considering blowing in a whistle, the high whistling sound is simply the transmission material air which vibrates a high amount of Hz. A tone played by a musical instrument consists of a duration, a loudness and a pitch, among other features [4]. The pitch is the part of the tone which is related to the amount of wave formed cycles per unit of time. Two tones with the same duration and loudness but with different pitches will sound different and each tone can be judged as either higher or lower.

Humans don't perceive sound as wave formed signals, but a computer does. From a wave formed signal one can find the amount of cycles per second the signal contains, also called the frequency of the signal. There are a couple of technical ways to establish the frequency. The most famous and well known is the Fast Fourier Transform (FFT) [5], which computes the discrete Fourier transform (DFT). The DFT is a complex-valued transformation describing the frequency content of the signal. The original wave formed signal is transformed by the DFT to the frequency domain. The formula of the DFT in [5] is:

$$X_k = \sum_{n=0}^{N-1} x_n e^{\frac{-i2\pi kn}{N}} \quad k = 0, ..., N-1 \tag{1}$$

where X is the output sequence (the Fourier transformed signal), x is the wave formed signal and N is the length of the wave formed signal. The FFT computes the DFT with less operations than calculating the DFT directly from the sum in (1), according to [5], and that is the reason why the FFT has become so famous. What practically happens in the DFT is that the data is divided into equally spaced frequency bins, and for each frequency bin a complex-value is calculated corresponding to a phase and amplitude of the frequency content in the signal.

The output of the DFT is complex-valued and its absolute value corresponds to the amplitude of the transformed signal. The absolute value gives an illustration of the signal's frequency content which is easier to interpret, and is called the amplitude spectrum. In Figure 1, a guitar playing the tone A3 is visualized together with the amplitude spectrum[1].

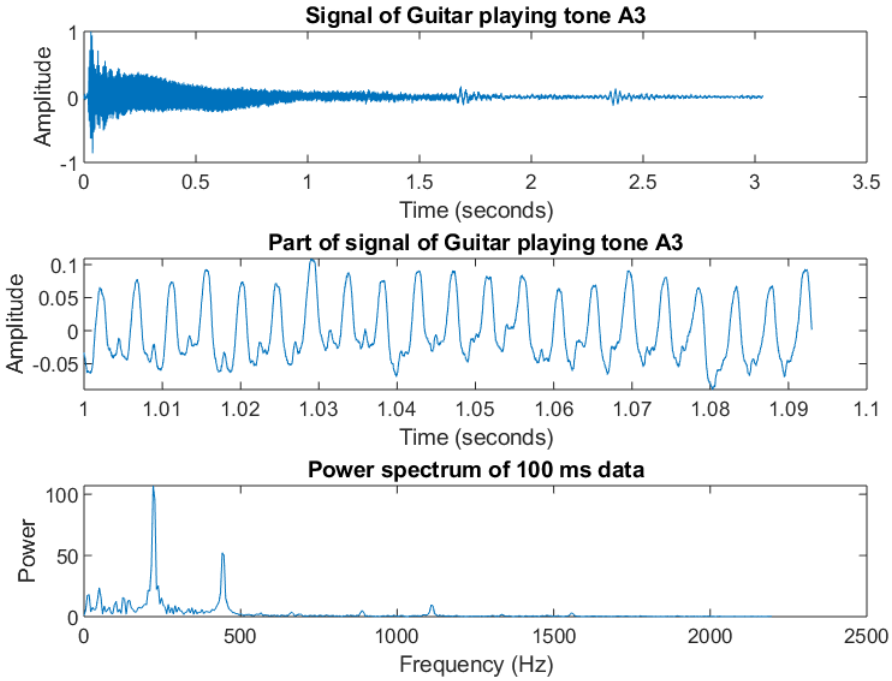

Figure 1: Top plot shows how the strike of the guitar string appears right after the time starts. One can also see how the power is decaying as the tone is fading out. The middle plot shows a zoomed in version of the above plot which is highlighting the wave form of the signal. The bottom plot shows the absolute value of the discrete Fourier transform which has a high peak at 220 Hz and a smaller but distinct peak at 440 Hz. By looking closer to the bottom plot one can find some smaller peaks at higher frequencies as well.

About the amplitude spectrum in Figure 1 some words must be said. The tone A3 corresponds to the pitch of 220 Hz, and a high power in the amplitude spectrum for that frequency can be noticed, corresponding to a high content of that frequency. Another distinct amplitude appears at 440 Hz which would correspond to the pitch of tone A4. A medium high content of the frequency 440 Hz is found even if tone A4 was not played. The peak at 440 Hz in the spectrum has appeared due to a property of musical instruments called overtones, as explained in [6]. Notice that the second peak appears around 440 Hz which is 2×220 Hz. This approximately multiple phenomena of the peaks is not a coincidence. By looking closer at the amplitude spectrum, one finds the smaller peaks in higher frequencies to occur at approximately multiples of 220 Hz (660,

[1]In music transcription one is interested in describing the signal as much as possible by the frequency representation. In order to perfectly recreate the signal from its frequency representation, the signal needs to be stationary. Clearly the top figure is not stationary, because of the changes in the envelope of the curve. If one studies a shorter segment of the data, like the middle plot, the data can be considered as stationary. The amplitude spectrum in the bottom plot is determined from 100 ms of data.

880, 1100 Hz and so on). All these multiples are called overtones. The frequency corresponding to the actual pitch is called the fundamental frequency. In many real-world sounds including signals from musical instrument, it is a common fact that overtones appear at approximately multiples of the fundamental frequency. The power in each overtone differs between instruments, as well as the envelope of a tone [6][2]. This is closely related to the concept of timbre which will be further discussed in this thesis.

One observation to notice from the amplitude spectrum in Figure 1 is that the fundamental frequency has more power than the overtones. The most powerful frequency is called the predominant frequency and the fundamental frequency is by far the most common predominant frequency. This discovery has contributed very much to the music transcription research [2].

The fact that fundamental frequencies appears as peaks in the amplitude spectrum together with the expectation of distinct amplitudes for frequencies at multiples of the fundamental frequencies, makes up the two main conditions for estimating fundamental frequencies. When the expected fundamental frequencies are found the pitches can easily be transcribed to tones in a music sheet. The so far mentioned theory is just a scratch on the surface of music translation, but it is basically enough to be able to understand the principles of music transcription. This process is summarized in Figure 2.

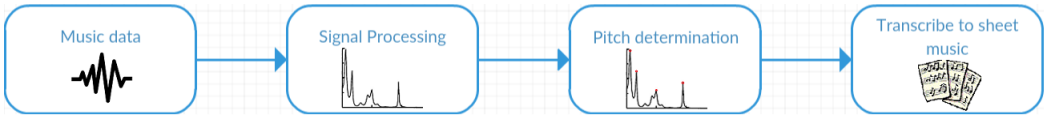

Figure 2: The main steps included in music transcription according to [2]. The word "Pitch" corresponds to the fundamental frequencies.

3.1.1 Spectral Analysis

To motivate the importance of advanced signal processing, one may return to Figure 1. Here the complicated wave formed signal of a guitar playing the tone A3 is transformed into an amplitude spectrum, where the sought fundamental frequency is easily seen. It should be mentioned that this is actually a very simple signal considering one instrument playing one tone. Further in this thesis the goal will be to estimate a polyphonic signal which consists of multiple tones, sometimes also played by multiple instruments. This causes new issues compared to estimatinga single tone from a single instrument.

First of all different instruments have different sounds. But also two instruments of the same kind may have different sounds. For example, two guitars may sound different because of the material of wood or strings. Actually a single instrument can have different sound as well, depending on how it is played and by whom. For example, one may play with different articulation such as staccato (detached) or legato (connected). This is included in a concept called timbre. Timbre is what distinguishes different sounds from each other and is a well investigated area. Different timbre can be exemplified by the envelope of the signal, the amount of noise, the power of the different overtones or how the overtones change through time (Spectral Envelope). The fact that a single pitch may have many different appearances, is a factor which makes polyphonic music

[2]The envelope of a tone is how the strength of the tone changes over time, for example from the strike of a guitar string until the sound fades out.

transcription a complicated problem.

Another issue in polyphonic music transcription is related to the earlier mentioned overtones, Section 3.1. By studying a simple example of 2 tones played by a guitar and the mixture of the tones, an interesting phenomenon occurs. The tones that are mixed is a guitar playing the tone A3 and a guitar playing the tone A4. The mixture is created by simply adding the signals together which is illustrated in Figure 3.

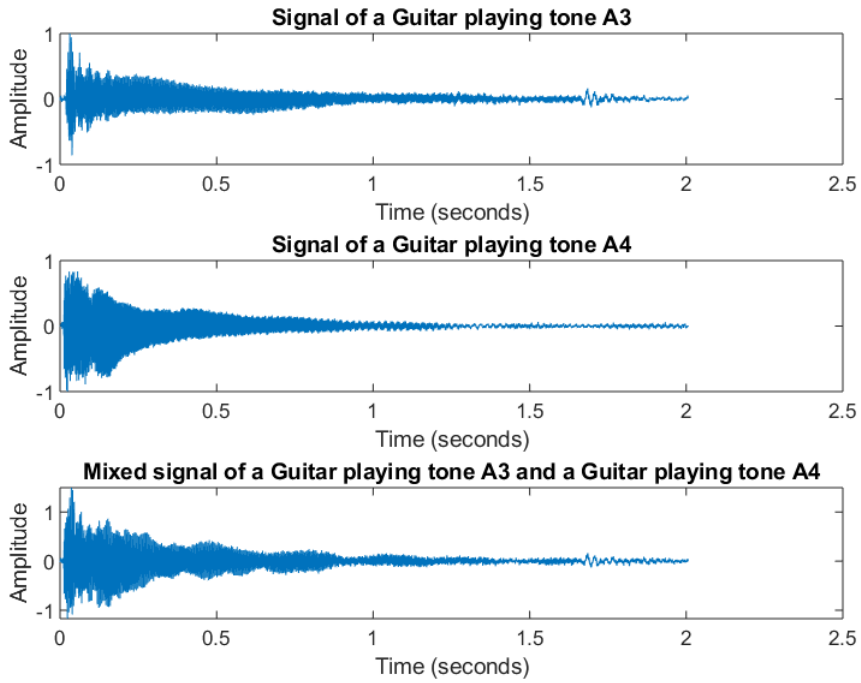

Figure 3: The top plot is the signal is the Guitar playing the tone A3, the middle plot is the Guitar playing the tone A4 and the bottom plot is the two signals added together.

The issue appears clearer by applying the FFT to these signals. This is illustrated in Figure 4.

From Figure 4 one finds a couple of interesting observations. By studying the top and middle plot one notices the similar structure of the overtones, which is a high peak at the fundamental frequency, 220 Hz and 440 Hz respectively followed by a distinct peak at the first overtone, around 440 Hz and 880 Hz respectively. In lower frequencies in the middle plot one finds a distinct peak corresponding to noise[3]. In the bottom plot there are three distinct peaks at 220, 440 and 880 Hz. The peaks at 220 and 880 Hz appear with similar amount of power as in the top and middle plot respectively. The peak at 440 Hz appear with more power than in both the above plots. This is explained as the first overtone of tone A3 overlapping with the fundamental frequency at A4. The plot can give the impression that the power of the peaks has been added together, which sort of has been the case here. But a different phase of the mixed signals could likewise cause the peaks to cancel each other out. To understand why this may happen, this thesis refers

[3]The noise could possibly come from AC voltage which has a frequency of 50 Hz and might affect the recording.

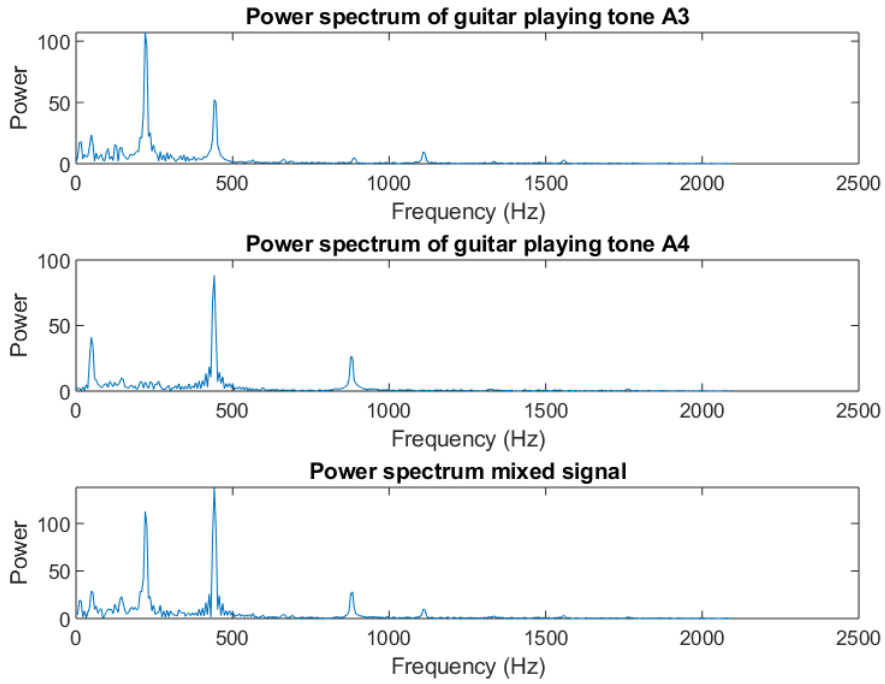

Figure 4: The top plot is the amplitude spectrum of a guitar playing the tone A3 (220 Hz), the middle plot is the amplitude spectrum of a guitar playing the tone A4 (440 Hz) and the bottom plot is the amplitude spectrum of the two signals added together. All amplitude spectrums in the plot are of 100 ms of data.

to theory about of addition of complex numbers [7]. Whether two overlapping peaks from two signals which are mixed together should be added together, take each other out, or anything in between, can in a real world scenario be considered as random. When dealing with polyphonic music signals, overlapping overtones will occur very frequently and as described, these overtones may not be easily identified or separated.

If one pretends to be given the bottom plot in Figure 4 and asked to determine the active pitches, one may think that it is an easy task just to pick out the two highest peaks. That would be easy given the number of active pitches and what instrument that is used. But when these questions should be answered automatically by a music transcription model, the number of pitches and what instruments used, are unknown. One may realize that this turns out to be a more complicated task. Given the knowledge that the piano has an overtone structure where the first overtone has about the same power as the fundamental frequency [6], the amplitude spectrum in the bottom plot in Figure 4, could likewise be estimated as a single tone played by a piano.

By adding more instrument signals to a mix, the number of overlapping overtones will just increase and the music transcription will appear more complicated. In order to have optimal conditions at the pitch estimation part (Figure 2) one would like to use some kind of spectrum which contains as much and robust information as possible and as little noise as possible. This is further discussed in Section 3.1.3.

3.1.2 Musical- and Overtone Theory

In this section some basic concept of music theory will be explained a bit further. Knowledge in basic musical theory is essential for a technically oriented music transcriber. In Section 3.1, it was stated that a musical tone consisted of for example a duration, a loudness and a pitch. In this section the pitch of the tone will be focused.

The pitch is the lowness or highness of a tone corresponding to a low or high amount of oscillating cycles per second in the wave-formed signal. Specific frequencies are named with a letter name corresponding to a certain tone, for example the pitch 440 Hz is called A4. The "4" in the name A4 means that it is a tone A in the fourth octave. A piano consists of tones from A0 at 27.50 Hz up to C8 at 4486.01 Hz which is illustrated in Figure 5

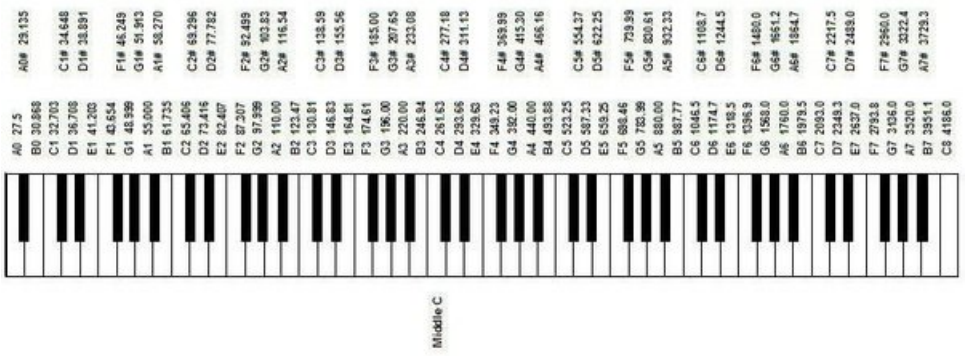

Figure 5: The piano keys with the corresponding letter and pitch in Hz [8].

An octave is a certain distance step between two tones of either the double or the half frequency. The tone A3 appears at 220 Hz and the tone A5 appears at 880 Hz which is half and twice the pitch of A4 correspondingly. Between each octave there are 12 different tones named with letters. The authors of [9] explain that the frequency of the tone which follows after A4 is $2^{1/12} \times 440 \approx 1.05946 \times 440 = 466.13$ and is named A$^{\#}$4 which is pronounced: A sharp four. It is also said that the distance between two adjacent tones is called a semitone and is always $2^{1/12} - 1 \approx 6\%$ of the frequency. Humans hear sounds in a logarithmic scale where the semitone distance sounds equal independently of what octave the tones are [4] [4]. Note that $440 \times (2^{1/12})^2 = 493.88$ which is B4 which is the tone after A$^{\#}$4. Also one may note that $440 \times (2^{1/12})^{12} = 440 \times 2 = 880$ which is A5. All frequencies from tone A0 to C8 are illustrated in Table 1. From the table it gets clear how the logarithmic structure of the fundamental frequencies of the tones appears.

In Section 3.1 the concept of overtones as something which occurs from musical instruments were introduced[5]. It was noted that overtones occurred at approximate multiples of the fundamental frequency. Considering the tone A3 with 220 Hz, the first overtone appears at approximately 440 Hz. According to Table 1, the same frequency appears as the fundamental frequency of the tone A4. The second overtone of A3 should appear around 660 Hz. One may note that there is no tone with fundamental frequency of 660 Hz in the table, but the tone E5 with 659.25 Hz is close.

[4]For example, the distance between A2 and B2 sounds equal as the distance between A3 and B3.

[5]The physical explanation of why they occur is not needed for further musical analysis.

	C	$C^{\#}$	D	$D^{\#}$	E	F	$F^{\#}$	G	$G^{\#}$	A	$A^{\#}$	B
0										27.50	29.14	30.87
1	32.70	34.65	36.71	38.89	41.20	43.65	46.25	49.00	51.91	55.00	58.27	61.74
2	65.41	69.30	73.42	77.78	82.41	87.31	92.50	98.00	103.83	110.00	116.54	123.47
3	130.81	138.59	146.83	155.56	164.81	174.61	185.00	196.00	207.65	220.00	233.08	246.94
4	261.63	277.18	293.66	311.13	329.63	349.23	369.99	392.00	415.30	440.00	466.16	493.88
5	523.25	554.37	587.33	622.25	659.25	698.46	739.99	783.99	830.61	880.00	932.33	987.77
6	1046.50	1108.73	1174.66	1244.51	1318.51	1396.91	1479.98	1567.98	1661.22	1760.00	1864.66	1975.53
7	2093.00	2217.46	2349.32	2489.02	2637.02	2793.83	2959.96	3135.96	3322.44	3520.00	3729.31	3951.07
8	4186.01											

Table 1: Frequencies of musical notes in the range of a piano. The notes in a scale are in each column and the octaves are in each row. The table should be read as the frequency of tone C2 is found in the C-column (1st column) on the 2-row (3rd row) as 65.41. The frequencies are rounded to 2 decimals.

Actually E5 is close enough in order to interact with the second overtone of A3 and one may say that A3 and E5 have overlapping overtones at around 660 Hz (interaction between overlapping frequencies were described in Section 3.1.1). Actually all tones in Table 1 with frequencies $\approx \frac{660}{n}$ for all $n \in \mathbb{Z}$ will have an overlapping overtone with A3 at 660 Hz. This similarly holds for all overtones of A3, also at the fundamental frequency. When describing all active frequencies in a tone, both the fundamental frequency and the overtones, at once, the term harmonics is used. The first harmonic is the fundamental frequency, the second harmonic is the first overtone and so on.

One may note that all combination of two tones, given enough active overtones, will sooner or later have two harmonics which will overlap[6]. For some combination of two tones this occurs for high frequencies while for others it occurs for lower frequencies. A mix of two tones where the first overlapping overtone appears at a high frequency are nicest from a music transcription perspective, since there are less interactions in the spectrum, which are hard to deal with. Unfortunately for the music transcriber, a mix of tones in music sound more pleasantly and sweetly from a psychological perspective if the tones are in harmony and the overtones become merged. This psychological feeling has been sought by music creators throughout time, and the music humans listen to is mainly in harmonies. It would have been advantageously for a music transcriber if the music humans listen to was out of harmony. But as the world look like, a user of a music transcription device would only be interested in transcribing harmonic music which sounds good, so that is what the music transcription models should be able to handle. The psychological phenomenon of good and bad sound according to different combinations of tones, is called consonance and dissonance between tones.

3.1.3 Signal Processing

FFT is this far the only signal processing method mentioned in this thesis. However, the FFT is rarely used in common music transcription models. The reason why it has been used in the above examples is partly because of its simplicity and its familiarity to technical skilled persons, but also since many of the other signal processing methods have similarities to the FFT.

[6]For two tones x and y, one can see that $x \times (2^{1/12})^m$ and $y \times (2^{1/12})^n$, for some $m, n \in \mathbb{Z}$, will be approximately equal.

One of the most frequently used signal processing methods in music transcription is the Constant-Q transform (CQT) [10]. To explain what it does it is often compared with the FFT, see (1). For the FFT, the frequency axis is linearly spaced, which means that the difference between two neighbouring frequency bins is always the same. In the CQT, the frequency axis instead is geometrically spaced. According to [10], its frequencies are

$$f_k = f_0 \cdot 2^{\frac{k}{b}} \quad k = 0, 1, ... \tag{2}$$

where b is the number of frequency bins per octave and f_0 is a chosen center frequency. The motivation of geometrically spaced frequency bins is that is more similar to the human logarithmic hearing. For example, the step difference between G3 (196 Hz) and A3 (220 Hz) does for a human sound the same as between G4 (392 Hz) and A4 (440 Hz) even if the step difference is twice as big measured in absolute frequency. The frequency resolution of the CQT remain constant in different octaves while the frequency resolution of the FFT is higher for higher octaves than lower octaves. This can also be considered as a downside of the CQT since the resolution of primarily higher frequencies are lost compared to the FFT. In the CQT, for each frequency f_k, the corresponding multiples $(...\frac{1}{4}f_k, \frac{1}{2}f_k, 2f_k, 3f_k, ...)$ will also get their frequency content evaluated, which is convenient due to the importance of overtones in musical transcription. For the FFT, the frequency axis will in general not contain all exact multiples of f_k.

About the parameters in (2), b is usually chosen to be a multiple of 12 since that is the number of tones in each octave. f_0 is usually chosen as the lowest tone which is expected in the signal (which for a piano would be A0 with 27.50 Hz). Because its better alignment to the human audio perception, the CQT (or versions of it) is considered to be a more robust transform than the FFT and is therefore more frequently used in music transcription models.

Other versions of the FFT where the frequency axis is differently scaled are commonly used in music transcription models. One variant is the Equivalent Rectangular Bandwidth where the frequency axis is transformed by

$$ERB(f) = 6.23 \cdot f^2 + 93.39 \cdot f + 28.52 \tag{3}$$

which is a designed to approximate the bandwidth of human auditory filters [11].

Another scaling of the frequency axis is by the Mel scale. Frequencies are transformed to mels m, by

$$m = 2595 log_{10}(1 + \frac{f}{700}) \tag{4}$$

which also is based on the human perception of sound [12].

3.1.4 Technical Complications

In a piano roll representation a tone is represented as a quarter note, a whole note or something similar given a rate of beats per minutes. Comparing a musician to a computer by playing a melody from a music sheet, the musician may hit all tones correctly, but due to tempo changes, different pronunciation or other timbre differences, they will not be identical. In order to create a dataset containing both a real music recording and a ground truth music sheet, the sheet music needs to be manually recreated from the recording. To create a such sheet music is not trivial. The music transcription models consider time bins of 10 ms, therefore the note sheet needs the

same accuracy. Considering a melody of 30 seconds, that is 3000 time bins which should be filled with the correct frequencies. Creating a perfectly working dataset is complicated and takes lots of time. This has caused a lack of established datasets, which is affecting the research.

The hardest part in manually recreating the sheet music is to determine where a tone ends, or specifically where the offset occurs. Due to the envelope of for example a guitar tone, with sharp fluctuations in the beginning and high power in the spectrogram, both the fluctuations and the spectrogram power decays towards offset of the tone[7]. Thus it is not perfectly clear where the tone ends. Often some kind of threshold is used when transcribing to determine when a tone should be considered active or inactive. By defining the onset and offset by a threshold usually gives an accurate precision of where the onset appears but a quite vague precision of where the offset appears.

Another issue when considering real recorded music is that the instrument must be tuned perfectly. If not, the pitches of the tones will differ from the ground truth. By recording a melody one needs to consider the choice of sampling frequency. The highest possible frequency which can be identified from a recording is the half sampling frequency, $\frac{f_s}{2}$, and is called the Nyquist frequency. The highest tone on a piano is the C8 which occurs at 4186.01 Hz. The sampling frequency should reasonably be of at least $2 \cdot 4186.01 = 8372.02$. A final technical consideration needs to be taken about the instruments which have a vibrating sound, like the flute or the violin, where the pitch tends oscillates around the true pitch. This is a feature which is not captured by the piano roll representation but may affect an estimation.

3.2 Pitch Estimation

This section will describe the step in the music transcription algorithm where a set of given spectrums are used to estimate the active pitches. One intuitive way of this would be to search for all the peaks is the spectrum, set a minimum threshold such that peaks caused by noise can be excluded, then finally transcribe the frequencies corresponding to the acceptable peaks into a music sheet. Using this method one would probably be able to find most of the correct tones, but since there will be peaks in the spectrums which correspond to overtones, a lot of incorrect tones will be estimated. Thus, a more sophisticated algorithm must be used.

By analyzing the models in the research field, there are in general three variants of pitch estimation methods: probabilistic, parametric and neural network-based methods. The probabilistic methods are complicated to explain in a general form but is based on combined estimated probabilities about features in the signal. Further a parametric method and a couple of neural network methods will be explained in detail. In general the different variants of methods ends up in probabilities for each pitch candidate to be a fundamental frequency, which is the pitch estimation part of the problem. Given those probabilities some optional method, called postprocessing, is used to choose which pitches should be the estimated fundamental frequencies, which completes the pitch estimation part.

3.2.1 Subharmonic Summation

In order to not mistakenly estimate overtones as fundamental frequencies one way is to use the Pair-Wise Subharmonic Summation [2]. This method compares all the peaks (potential pitches)

[7] Also called the onset of a tone.

in the spectrogram with each other. For each pair of the pitch candidates there will be one higher and one lower frequency which are notated by f_{high} and f_{low}. [2] divides the algorithm into two parts, and first it is assumed that they are successive harmonics. If the ratio $h = \frac{f_{low}}{f_{high} - f_{low}}$ is close to an integer then h can potentially be the harmonic number of f_{low} and correspondingly $h + 1$ can be the harmonic number of f_{high}. As in [2], similarly it is tested if the peaks have an odd harmonic relation by $h = \frac{2f_{low}}{f_{high} - f_{low}}$. If the ratio is close to an integer, h can potentially be the harmonic number of f_{low} and correspondingly $h + 2$ can be the harmonic number of f_{high}. Following the algorithm in [2], after all possible pairs of peaks has been evaluated, f_p, called the virtual pitches, are calculated as $f_p = \frac{f_i}{h_i}$. The virtual pitches correspond to weights which after a few more steps in the algorithm in [2], are added to the peak candidates of the fundamental frequencies.

There are a couple of ways these candidates can be classified as pitches. The simplest way is by setting a straight threshold which may have been achieved through training on data. The pitch candidates who fulfill the threshold condition becomes the estimated fundamental frequencies. There are more advanced variants of thresholds which will be described in Section 3.2.5.

The subharmonic summation is a method which goes under the category of parametric methods since the weights are threatened with trained parameters in order to form the pitch candidates. The subharmonic summation is built on the assumption that overtones always appear in music signals, which is most often true. But a weakness is that different instruments have different amount of overtones which might favor or disadvantage different musical instruments. The subharmonic summation was in 2016 claimed by [2] to be the best and most current method of pitch estimation.

3.2.2 Neural Networks

Neural networks have recently become immensely popular with an ever increasing number of applications. Before 2016, the use of neural networks in music transcription was very limited but nowadays the neural network methods are dominating the field[8].

Neural networks are complicated systems loosely inspired by the human brain. By feeding the network with a lot of training data one can explain a neural network as a system that learns itself to find patterns and to solve problems. Classification problems are typical problems which neural networks have been proven to be successful within. To reconnect this to music transcription, one can describe the pitch estimation as a classification problem where some input data of the spectrum form is given. Each of the potential pitches (for example the 88 tones for a piano from A0 (27.5 HZ) to C8 (4186.01 Hz)), should be estimated as either active (1) or inactive (0).

There are many different types of neural networks, but a few are more popular in the musical transcription area. The most common network types are the convolutional neural network (CNN) and different types of recurrent neural networks (RNN) such as the long-short term memory (LSTM) network. This section describes how a standard neural network, called a Multilayer perceptron (MLP), works with theory based on [13], [14] and [15]. Further, in Section 3.2.3, a review of different types of networks will be presented.

[8]Both according to the participating models in the music transcription competition MIREX and other publications in the area.

Every neural network consists of an input layer, one or more hidden layers and an output layer [14]. The input layer consists of neurons which is explained in [14] as a holder of a number which is determined by what is fed into the network. In a typical neural networks, a vector of data is fed to the network which can be of various types, e.g., time series data, multidimensional biological data or images. Not all of these seem naturally as a vector of numbers, but by stacking multiple columns or transforming colors by the RGB-scale for example, a vector can be created and used as neurons in the input layer. The output layer does also consist of neurons and the amount of neurons in the output layer is usually specified to the number of classes there are in the data [14]. For example, if one wants to classify images of hand written digits one naturally specifies 10 classes each corresponding to the numbers 0-9. The hidden layers are described in [14] as the heart of the neural network. For an MLP there can be one or more hidden layers. The hidden layers do also consist of neurons, called hidden units. The numbers the hidden units consist of are values in the input layer and some weights [14]. The MLP is a fully connected network, which means that all neurons in the input layer are connected to each of the neurons in the first hidden layer, and each of these connections have an individual weight [13]. From the hidden units in the first hidden layer there are connections to the hidden units in the second hidden layer and so on such that all neurons are connected to all neurons in the other layers over the whole network[9]. The connections from the input layer to the first hidden layer is mathematically written in [13] as:

$$a_j = \sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}, \quad j = 1, ..., M^{(1)} \tag{5}$$

where a_j is called the activation of the j'th hidden unit. The superscript, (1), indicates that the parameters correspond to the first hidden layer which consists of totally M hidden units[10]. x is the data in the input layer which consists of totally D neurons. $w_{ji}^{(1)}$ are the weights, each corresponds to the connection between a neuron in the input layer and a neuron in the hidden layer. $b_j^{(1)}$ are called biases and can be seen as constant terms. The weights and biases can both be positive and negative, and initially they are randomly initialized for example by drawing from a Gaussian distribution [13].

The hidden units z are, as in [13], formed by a differentiable activation function $h(\cdot)$

$$z_j = h(a_j) \tag{6}$$

where a_j are the activations from (5). The activation function can for example be chosen to be a sigmoidal function, such as the *logistic sigmoid*

$$z_j = \frac{1}{1 + e^{-a_j}} \tag{7}$$

which squeezes $\mathbb{R}$ onto $[0, 1]$, as defined in [13], or the *tanh* function. Another popular activation function is the *Rectified Linear Unit* (ReLU), also defined in [13], which is determined as

$$z_j = max(0, a_j). \tag{8}$$

[9]Note that a neuron never is connected to another neuron in the same layer.

[10]That the parameters correspond to the first hidden layer means the connections from the input layer to the first hidden layer.

For the second hidden layer, in [13], the activations are formed similarly to the first hidden layer, by

$$a_k = \sum_{j=1}^{M^{(1)}} w_{kj}^{(2)} z_j + b_k^{(2)}, \quad k = 1, ..., M^{(2)} \tag{9}$$

Where the superscript, (2), indicates that the weights and biases correspond to the second hidden layer. The hidden units z_j corresponds to the first hidden layer which was determined in (5) and (6). To achieve the hidden units of the second hidden layer an activation function as in (6), is applied to (9).

This continues in the same way for all hidden layers until the output layer appears. For the output layer, the activation function often differs from the functions which are used for the other layers, and is denoted by σ instead of h. The output layer activation function is determined by the purpose of the neural network. If the purpose is linear regression, then the activation function could simply be $y_k = \sigma(a_k) = a_k$. If the purpose is classification, the softmax function

$$y_k = \sigma(a_k) = \frac{exp(a_k)}{\sum_j exp(a_j)} \tag{10}$$

as defined in [13], could be used. If one has a neural network with a single hidden layer, the output can directly be written as a function of the input, i.e.,

$$y_k(x, w) = \sigma\left(\sum_{j=1}^{M^{(1)}} w_{kj}^{(2)} h\left(\sum_{i=1}^{D} w_{ji}^{(1)} x_i + b_j^{(1)}, \right) + b_k^{(2)} \right), \quad j = 1, ..., M^{(1)}, \quad k = 1, ..., M^{(2)} \tag{11}$$

where $w_{kj}^{(2)}$ and $b_k^{(2)}$ are the weights and bias between the hidden layer and the output layer. $M^{(2)}$ is the number of neurons in the output layer. The MLP consists basically of input data, activation functions and weights. The format of the input data and the variants of activation functions are manually designed when creating the neural network. The weights have to be determined by training which is done by feeding the network with training data. For supervised learning the output is known and the weights are adapted by a computer to minimize an error function. The error function may be chosen with regards to the purpose of the network. For many common applications of a neural network, the error function, $E(w)$, may have the form as in [14], which is

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} (y(x_n, w) - t_n)^2 \tag{12}$$

called squared loss error, where w are the weights, x_n is the n'th input vector of totally N training samples, $y(x_n, w)$ is the estimated output by the network (similar to (11)), and t_n is the expected output which is given in the training data, also called the ground truth. The error function will for each training example achieve a cost and by taking the average cost of all the training samples one will get a score on how well the network is doing [14]. A low cost is desirable and by using gradient information, the optimizer may change the weights in order to find a lower cost value. It should be noted that this is a quite complicated procedure since this single error score should be used to optimize all the weights in the network, which for a reasonable simple

neural network structure, still can be tens of thousands of weights. It is described in [14] that this huge amount of weights makes the error function extremely multidimensional, which results in a large amount of minimum points. There are ways to analytically find a local minimum but not a global one [15]. In order to find the optimal minimum of the error function, the global minimum needs to be found, which is a complicated task. The strategy of finding the global minimum is closely related to the way of finding local minimum points.

Local minimum points of a function in a multidimensional space can be found where the derivative is equal to zero. For that assumption to hold it is of great importance that the function is differentiable over the whole space [13]. From (11), considering differentiable choices of activation functions, h and σ, one sees that neural networks are differentiable. A differentiable function $y_k(x_n, w)$ inserted into (12), the error function is also differentiable.

Given some starting point in the multidimensional space, there are different ways of finding local minimum points. In [15] Conjugate directions method, Newton's method or Steepest descent are mentioned as methods which are used for this kind of task, but with some differences in efficiency and assumptions of once or twice differentiable functions. The algorithm of the steepest descent, described in [15], which assumes a once differentiable function method will further be explained herein. The derivative at the starting point x is first calculated. Since $\nabla y(x)$ indicates in which direction y has the largest growth, $-\nabla y(x)$ indicates in which direction y has the steepest descent. Steepest descent then makes a one-dimensional search along this direction

$$d = -\nabla y(x) \tag{13}$$

for the minimum point. The process is then repeated. This is thus an iterative method, where the next point is determined by

$$x_{k+1} = x_k + \lambda_k d_k \tag{14}$$

where λ_k is chosen such that it satisfies, for example, the Wolfe conditions or the Barzilai-Borwein method [16], which guarantee that $y(x_{k+1}) < y(x_k)$. By letting $k \to \infty$ the minimum value will be found under certain conditions. In practice one will not do this infinitely many times. Instead [15] solves this by setting a threshold value close to zero, and if the derivative is below that threshold, a minimum value is assumed to be found. [15] claims that this method actually turns out to be poor in relation to the other mentioned methods. Further it is explained that the steepest descent especially has problems when the function's curvature is of an elliptic shape. The reason why this example was used is because the other ones have similar structure but are less intuitive.

By the steepest descent method a local minimum can be found[11]. Considering an extremely multidimensional function, as the error function of a neural network, a random local minimum will not be a good choice of weights, since the probability that it would be the global minimum is extremely low, because of the large amount of local minimum points [14]. A very ineffective way would be to find thousands of local minimum points by randomizing starting points, and then choose the best one. Anyhow, this approach would not necessarily give a good result [15]. A cleverer way of updating the weights in a neural network is therefore needed. The regular way of doing this is through a method called backpropagation.

[11] Even if another choice of method would be more appropriate.

In backpropagation the weights are being estimated by starting from the back of the network (the output layer) and then moving backward to end up with the weights from the input layer. In [14] this is described by zooming in on the cost of a single weight, w_{jk}, between the k'th output neuron y_k and the j'th neuron from the last hidden layer z_j in a single training example x_0. The cost function from (12) appears as

$$C_k = \frac{1}{2}(y_k(x_0, w_{jk}) - t_{k_0})^2 \tag{15}$$

where t_{k_0} is the ground truth of class k of a single training example, denoted by subscript 0. C_k is the cost of this specific example[12]. $y_k(x_0, w_{jk})$ is the estimated class k output value of the network. Note that this is a clear simplification since in a standard neural network y_k would depend on w_{jk} for all j and not just a single weight. In the optimal case $C_k = 0$. If C_k is large it means that the system needs to put much effort in changing $y_k(x_0, w_{jk})$. If C_k is close to zero the system will not put as much effort in changing it. In [14] it is shown that $y_k(x_0, w_{jk})$ can be rewritten as a function of the earlier steps in the network by

$$y_k(x_0, w_{jk}) = \sigma(w_{jk}^{(L)} z_j^{(L-1)} + b_j^{(L)}) \tag{16}$$

where w_{jk} is the weight and b_j is the bias. (L) indicates the last layer (the output layer) and the $w_{jk}^{(L)}$ and $b_j^{(L)}$ are connections between layer $L-1$ and layer L. Here, σ is the activation function and z_j is the j'th hidden unit of layer $L-1$. It is possible to rewrite this further as $z_j^{(L-1)} = h(w_{ij}^{(L-1)} z_i^{(L-2)} + b_i^{(L-1)})$ and so on where i is the i'th hidden unit of layer $L-2$. In order to keep this intuitive the form of (16) will be used. In backpropagation it is of interest to change $y_k(x_0, w_{jk})$ depending on how much error it causes [14]. This is determined by calculating weights' and biases' impact on $y_k(x_0, w_{jk})$, since the weights and biases is what indirectly influence the cost of $y_k(x_0, w_{jk})$. The impact of the weights is calculated by the derivative $\frac{\partial C_k}{\partial w_{jk}^{(L)}}$ [14]. Recall that in backpropagation the training starts from the back, at layer L, which means, first estimate $w_{jk}^{(L)}$ (for all j and k), then $w_{ij}^{(L-1)}$ (for all i and j) and so on.

By the chain rule the derivative may, as in [14], be rewritten as

$$\frac{\partial C_k}{\partial w_{jk}^{(L)}} = \frac{\partial a_k^{(L)}}{\partial w_{jk}^{(L)}} \frac{\partial y_k}{\partial a_k^{(L)}} \frac{\partial C_k}{\partial y_k} \tag{17}$$

Where $y_k(x_0, w_{jk})$ is shortened as y_k and $a_k^{(L)} = w_{jk}^{(L)} z_j^{(L-1)} + b_j^{(L)}$ which is the inner part of (16). The derivatives of the right hand side of (17) can, as in [14], be calculated one by one as

$$\frac{\partial C_k}{\partial y_k} = y_k(x_0, w_{jk}) - t_{k_0} \tag{18}$$

$$\frac{\partial y_k}{\partial a_k^{(L)}} = \sigma'(a_k^{(L)}) \tag{19}$$

$$\frac{\partial a_k^{(L)}}{\partial w_{jk}^{(L)}} = z_j^{(L-1)} \tag{20}$$

[12] A more appropriate variable name would have been $C_{k_0, x_0, w_{jk}}$ but in order to simplify notation it is shortened as C_k.

where a couple of constant terms has been cancelled. Combining (18), (19) and (20), as in [14], makes (17) to be rewritten as

$$\frac{\partial C_k}{\partial w_{jk}^{(L)}} = y_k(x_0, w_{jk}) - t_{k_0})\sigma'(a_k^{(L)})z_j^{(L-1)} \tag{21}$$

which may be evaluated directly. One interesting fact from this is that the impact of the weights between layer $L-1$ and L is directly related to neuron $z_j^{(L-1)}$ in the hidden layer $L-1$. This means that a weight $w_{jk}^{(L)}$ only can be influential if the neuron z_j in hidden layer $L-1$, is influential.

By going further in the backpropagation and examining $\frac{\partial C_k}{\partial w_{ij}^{(L-1)}}$, one needs to take into account all the different paths each neuron can influence each output. For the neurons in layer $L-1$, the term $\frac{\partial C_k}{\partial y_k}$, may as in [14], be replaced by

$$\frac{\partial C_k}{\partial z_j^{(L-1)}} = \sum_{i=1}^{n_{(L)}} = \frac{\partial a_i^{(L)}}{\partial z_j^{(L-1)}} \frac{\partial z_i^{(L)}}{\partial a_i^{(L)}} \frac{\partial C_k}{\partial z_i^{(L)}} \tag{22}$$

which looks a bit messier, but it's actually just adding all connections together. The backpropagation in [14], for the weights, continues like this all the way back to the input layer. The same thing is done for the biases

$$\frac{\partial C_k}{\partial b_j^{(L)}} = \frac{\partial a_k^{(L)}}{\partial b_j^{(L)}} \frac{\partial y_k}{\partial a_k^{(L)}} \frac{\partial C_k}{\partial y_k} \tag{23}$$

where $\frac{\partial a_k^{(L)}}{\partial b_j^{(L)}} = 1$, which gives the following cost derivative

$$\frac{\partial C_k}{\partial b_j^{(L)}} = (y_k(x_0, w_{jk}) - t_{k_0})\sigma'(a_k^{(L)}) \tag{24}$$

which is found to be independent to other weights and biases in the L'th layer. The further steps in the backpropagation for biases is calculated in the same way as for the weights. The total cost of the whole network is in [14] defined by

$$C_0 = \frac{1}{2} \sum_{h=1}^{N} \sum_{l=1}^{n_{(L)}} (y_l(x_h, w) - t_{l_h})^2 \tag{25}$$

where N still is the number of training samples, $n_{(L)}$ is the number of neurons in the output layer, $y_l(x_h)$ is the l'th neuron in the output layer (determined by w_{jl} for all j), x_h is the h'th input, l_h is the corresponding expected output, and w indicates all weights and biases in the network. The cost derivative is calculated for all weights and biases and suggests how much each of them should change to get a better estimation. Combining the cost derivatives one may, as in [14], form

$$\nabla C_0 = [\frac{\partial C_0}{\partial w^{(1)}}, \frac{\partial C_0}{\partial b^{(1)}}, ..., \frac{\partial C_0}{\partial w^{(L)}}, \frac{\partial C_0}{\partial b^{(L)}}] \tag{26}$$

where

$$\frac{\partial C_0}{\partial w^{(1)}} = [\frac{\partial C_0}{\partial w_{11}^{(1)}}, ..., \frac{\partial C_0}{\partial w_{1n_{(1)}}^{(1)}}, ..., \frac{\partial C_0}{\partial w_{n_{(2)}1}^{(1)}}, ..., \frac{\partial C_0}{\partial w_{n_{(2)}n_{(1)}}^{(1)}}] \tag{27}$$

are the cost derivatives for each connection weight between the input layer and the first hidden layer. The bias derivatives

$$\frac{\partial C_0}{\partial b^{(1)}} = [\frac{\partial C_0}{\partial b_1^{(1)}}, ..., \frac{\partial C_0}{\partial b_{n_{(2)}}^{(1)}}] \tag{28}$$

are similarly calculated as the weight derivatives. The cost derivative is calculated for all the training samples[13]. This gives suggested changes for each of the weights in order to minimize the cost function by averaging the derivatives over for all the training data [14]. Then a local minimum point is search for around the suggested weights. This is done by one of the minimum searching methods and then a new round of backpropagation is applied. These two alternates an optional (preferably large) amount of times until a local minimum can be considered close enough a global minimum [14].

The statement that the cost derivative was calculated for all the training samples needs to be clarified. However, calculating the derivative from all training samples is expensive, and typically not done in practice. Instead [14] says that all training data usually are divided into mini-batches which each consists of a smaller amount of randomly chosen training data. The derivative of a mini-batch only approximates the derivative of the whole training set, but it greatly reduces the computational complexity. [14] claims that letting the network train to the limit, it will converge to the same answer but much faster.

3.2.3 Types of Neural Network

The MLP neural network described in Section 3.2.2 is the simplest version of a feed forward neural network. Feed forward means that the connections between the nodes do only send information forward in the network, which means that no cycle connections exists. A commonly used variant of feed forward neural network is the convolutional neural network (CNN) which is introduced in this section. Moreover, a neural network which is not categorized as feed forward may use recurrent layers. Such occur in recurrent neural networks (RNN) and its extension called long short-term memory (LSTM) networks which are mentioned later in this section.

The CNN is the most common architecture used for classifying objects in images. To this effect, [17] mentions classification of the MNIST-dataset, containing hand written digits, as an example where the CNN has been proved successful. The CNN is specifically designed to find local structures and to combine them, or in other words it is said to be shift invariant, which means that it doesn't matter where in the image the object is located [17].

The CNN has an input layer, an output layer and multiple hidden layers just as the standard MLP neural network and according to [17] the input is commonly an image with a certain size (it is important that this size is constant for all images). The output is commonly a couple of nodes, each corresponding to a probability of a certain classification. If a neural network consists of a convolutional layer it is said to be a CNN which is what differs from an MLP [17]. A standard CNN commonly also consists of pooling layers and fully connected layers.

[13] Actually not all samples at the same time. This will be discussed further down.

An image is just pixels with numbers in different scales (for example RGB-scale or black and white scale) and these numbers of $n \times m \times 3$ (RGB-scale) or $n \times m$ (Black and white scale) are what the input neurons are filled with. The first layer in a CNN is generally a convolutional layer [17]. In the convolutional layer one starts by considering an optional small area in the top left of the image, for example 5×5 pixels. The convolution is made by multiplying each pixel in the area with a specific weight, and then summing all these weights up, the area is estimated into one number which is inserted to a corresponding neuron. The next step is to move the area of 5×5 pixels a specified number of steps to the right (this number is called strides) and then repeat the convolutional steps. This continues until one reaches the right border of the image[14][17]. When the top row is finished, the area of 5×5 pixels moves a specified number of steps down from the top left corner and continues in the same way until the bottom right corner is reached [15] [17]. The layer which follows from a convolutional layer can be described as a new image, which now has a new size according to the specified settings of the pixel area, strides and padding.

A CNN or other neural networks consisting of many hidden layers are called deep networks. After a convolutional layer, a ReLU layer and some kind of pooling layer generally follows, [17]. The ReLU layer is a non-linear activation function which truncates negative values to zero, as in (8). The pooling layer is a kind of non-linear down-sampling where a specified area of an image is convolved by its max value or average value. It can be noted that the ReLU and pooling layer don't involve any weights. The pooling layer does only simplify the training of the networks, while the ReLU makes the communication in the network possible.

The network structure of a CNN does typically rotate by a convolutional layer, a ReLU layer and a pooling layer [17]. Typically, the last layer is fully connected and connects each pixel in the second to last layer to each of the outputs, as explained in Section 3.2.2. Since the CNN is a feed forward network, it is trained through backpropagation, similarly as the MLP.

One may wonder why a CNN is considered in a thesis about music transcription, when CNN mainly is applied to image classification. By considering the participants in MIREX 2016-2017 [18], CNN is actually a very commonly used method in the music transcription area. By stacking spectrums for consecutive time steps, a time-frequency representation with the power in the third dimension is obtained, called the spectrogram, and is visualized in Figure 6. Analyzing a spectrogram image means analyzing spectrums over time. This seems to be a reasonable approach since a spectrum is highly dependent of the adjacent spectrums.

Another type of neural network layer is the recurrent layer, which in its simplest form occurs in the RNN[16]. An RNN is similar to an MLP in the sense that the layers contribute to the following layers through an activation function with weights. The difference is that an RNN uses the fact that the input is formed as a sequence, where the next value in the sequence is related to the previous ones [19]. If the input is related to the previous inputs in the sequence, then the activations must be related to the previous activations in the sequence. To use this relation, the

[14]The right side of the right boarder is padded, which means that zeros are added outside the image. When the right boarder of the image is reached, there are two standard options, either to stop just as the padded zeros are reached, or one may continue until the 5×5 pixel area just contains the right boarder of the image. Which one to use is optional but the following layer will have different size depending of the choice.

[15]The bottom side of the image is also padded and the same boarder options occurs for the bottom boarder as for the right boarder.

[16]Recurrent neural network.

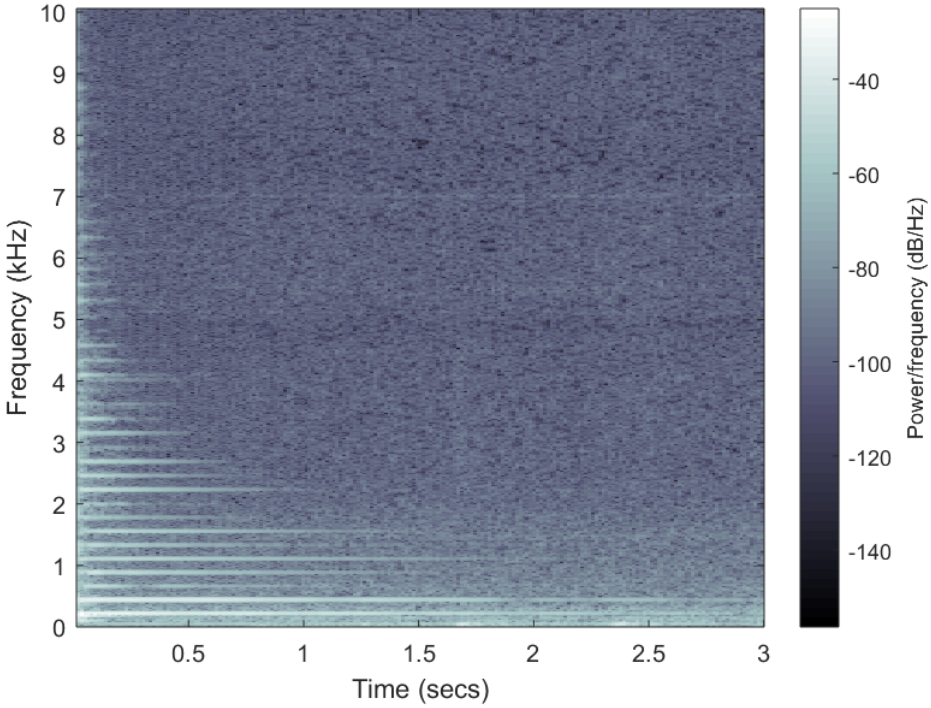

Figure 6: Spectrogram of a tone A3 played by a guitar. One may find that the power changes over time, but the changes are quite smooth. By considering two adjacent spectrums with 10 ms (0.01s) difference, the power doesn't change very drastically.

activations are saved for the following sequence step. The activation at the second step in the sequence is therefore dependent of both the input data from the second step in the sequence and the activation from the first step in the sequence. The connection schedule is shown in Figure 7.

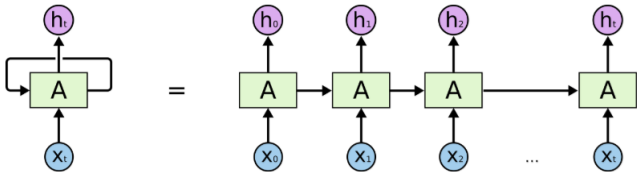

Figure 7: Recurrent neural network where x_t is the input at sequence t and h_t is the activation at sequence t. The activations move both forward in the network and as input to the next activation. The image is from [19].

For an RNN, the input data is always a sequence of vectors of size $n \neq 0 \in \mathbb{N}$, [17]. For a recurrent feature to make any sense, one finds that the data at step t in the sequence needs to have some relation to the data at the steps $[0, t-1]$ in the sequence. One example of this is of course time series data, where the specific example of spectrums from succeeding time bins is applicable in music transcription. MNIST is a typical dataset for which the recurrent feature in an RNN would not be of any use. This is because the order of the samples of hand written digits is randomized.

[17] If the input data would not form a sequence, for example if the input would be a single vector, the recurrent feature in the recurrent layer would be unused and the RNN would be identical to the MLP.

For an RNN the weights don't change for different steps in the sequence. The algorithm described in [19] for an RNN, is the following:[18]

$$h_t = w_1 x_n + w_2 h_{t-1} \tag{29}$$

where w_1 and w_2 are two different weight matrices. x is the previous layer which for example could be the input, and h is the activations. This gives

$$h_0 = w_1 x_0, \quad h_1 = w_1 x_1 + w_2 h_0 = w_1 x_1 + w_2 w_1 x_0 \tag{30}$$

and similarly

$$h_2 = w_1 x_2 + w_2 h_1 = w_1 x_2 + w_2(w_1 x_1 + w_2 w_1 x_0) = w_1 x_2 + w_2 w_1 x_1 + w_2^2 w_1 x_0. \tag{31}$$

Estimating parameters for an RNN is typically quite complicated, [19]. For the RNN, a variant of backpropagation, called backpropagation through time, is used in [19]. The problem is that backpropagation through time needs to handle a structure of dependencies. Since backpropagation through time is built on the derivatives of dependencies, the influence of x_0 for the activation h_{100} will contribute with $w^{100} \cdot x_0$, which is critical. If $w < 1$ the contribution of x_0 will be multiplied with almost nothing ($0.9^{100} = 0.00002656139$) and if $w > 1$ the contribution of x_0 will be very significant ($1.1^{100} = 13780.6123398$). This relatively small change of $1.1 - 0.9 = 0.2$ for the weights can make a considerably large difference for the output. Many local minimums will appear due to these large influence possibilities of the weights but the most of them will give very bad estimates [19].

This issue is called the vanishing or exploding gradient, and in order to solve it, the RNN is often extended. One extension is the LSTM network, which builds on the idea that all information from all previous sequences is not needed to be used, only the previous information which turns out to be important should be used[19]. In a LSTM, the network learns which information that can be useful for later and the rest is thrown away. The structure of the LSTM network is shown in Figure 8.

The LSTM has an input layer and ends up by calculating an activation, just as the RNN. Also, the activation of the previous sequence is used as input for the next. The main difference described in [19], is that there is another input from the previous sequence as well, called the cell state C. The cell state is visualized in Figure 9.

It is the cell state which holds the important information which will be saved for many sequences in the network. Information in the cell state can be added or deleted through each sequence. What information that is added or deleted is determined through weights trained by the network.

In [19], a forget gate layer f_t is created through $f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$ where W_f is the corresponding weight matrix, b_f is the corresponding bias, and σ is a sigmoid, (7), which forces all values to be between $[0, 1]$ [20]. The forget gate layer is convolved with the previous cell state. The f_t values which are close to 1 makes the cell state to remember its previous value while a f_t

[18] The algorithm can be easier understood by studying Figure 7.

[19] LSTM stands for Long short term memory.

[20] Just as for other neural networks, the weights and biases are the only parameters which are trained by the neural network.

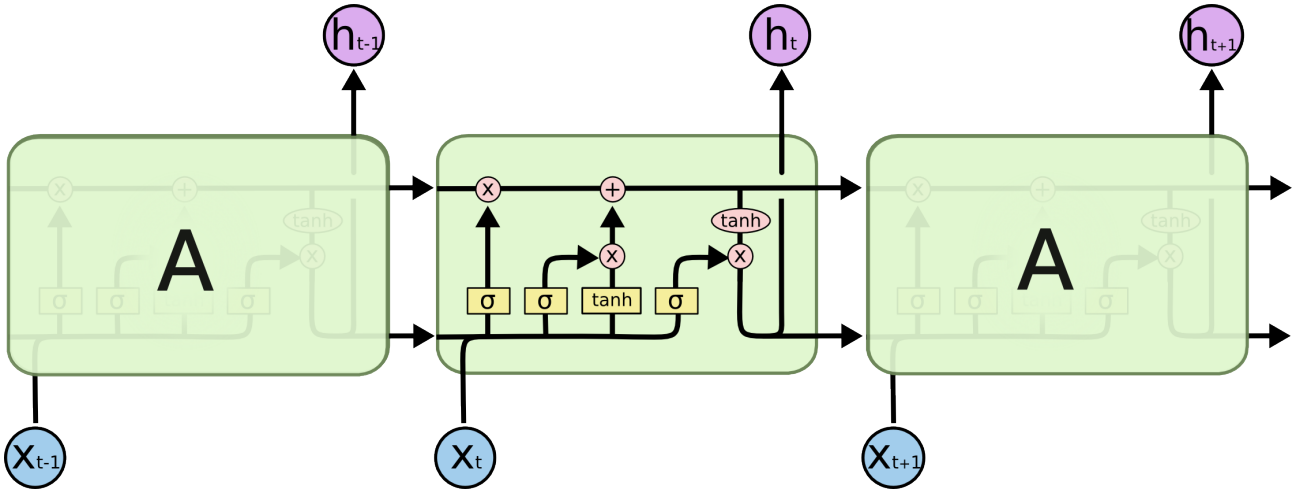

Figure 8: The structure of a Long Short-Term Memory network. The image is from [19].

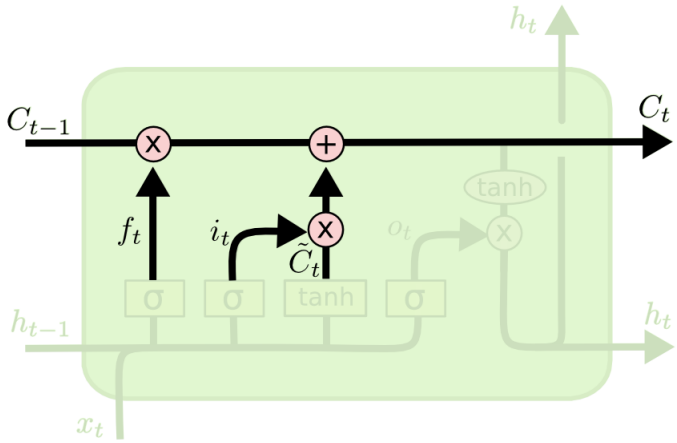

Figure 9: Close-up of the cell state, C_t, which is the long term memory part of the Long Short-Term Memory network. The image is from [19].

value close to 0 makes the cell state to forget its previous value.

The next step in [19] is to add new information to the cell state, which is done in two steps. First the input gate layer $i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ decides which values in the cell state that should be updated, where W_i and b_i are the weights and bias corresponding to the input gate layer. The second part is to create new cell state candidates $\tilde{C}_t = tanh(W_{\tilde{C}}[h_{t-1}, x_t] + b_{\tilde{C}})$ [19]. $W_{\tilde{C}}$ and $b_{\tilde{C}}$ corresponds to weights and bias for the candidate values. Here the *tanh* activation function maps all values to $[-1, 1]$. The input gate values are then convolved with the candidate values and then added to the cell state. The cell state finally becomes

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{32}$$

and will be sent to the next sequence as well as it will be used for creating the activations of the current sequence. The activations are created both through the activations from the previous sequence h_{t-1}, the input layer x_t and the current cell state C_t. How it's done is illustrated in Figure 10.

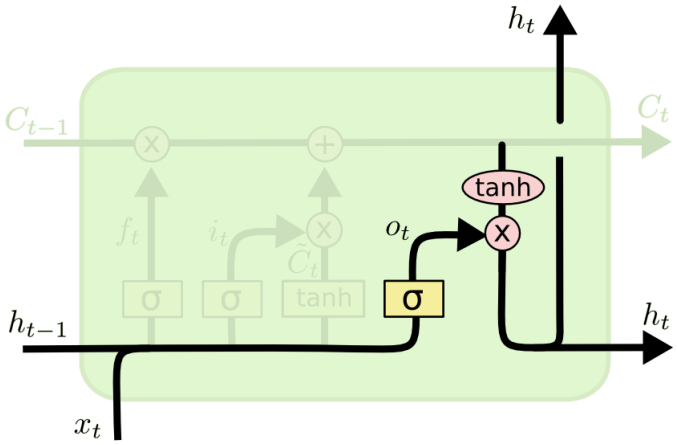

Figure 10: Close-up at the output layer which is the short term part of the Long Short-Term Memory network. The image is from [19].

The output layer $o_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$ in [19] corresponds to the short term memory part of the LSTM, with W_i and b_i as corresponding weights and bias. The output layer is multiplied with a filtered version of the cell state using the *tanh* activation. The final activation in [19] then becomes $h_t = o_t * tanh(C_t)$.

The weights and biases are determined from backpropagation through time. One may note that the issue of vanishing or exploding gradients have been stabilized since h_t only do additive updates of C_t instead of multiplicative as for RNN. The parameter estimates for a LSTM thus become more stable [19]. This is one reason why the LSTM is more applicable than the RNN, both in music transcription and other areas.

3.2.4 Complex Neural Network

Neural networks are often described as a computer version of the human brain with millions of neurons which communicates between themselves through connections. Often are these neurons described either as active 1 or inactive 0. This might be a reason why complex-valued neural networks haven't been a high topic area of the current neural network research. Recently it has been found that complex-valued neural networks can contribute in many areas both for image classification, music transcription and speech recognition[1, 20][21].

All theory of neural networks described herein, are so far adapted for real-valued numbers. How these real-valued operations and representations should be adapted for complex values are not obvious and currently (spring 2018) none of the open source neural network libraries supports complex-valued numbers. In this section, complex-valued counterparts to the convolutions, the activation functions and batch normalization are introduced with the theory based on [20].

First a short introduction to complex values is given. A complex number is written as $z = a + ib$,

[21] The autors of [1] very recently, July 2018, achieved 100% rate on the MNIST dataset using complex-valued neural network. Another example is [20] who used Deep complex-valued neural networks on all these areas with impressive results.

where a is the real part and b the imaginary part, where $i^2 = -1$. In order to make it compatible with the non-supporting neural network libraries, z is in [20] represented as a $n \times 2$-matrix with a and b in each column.

A complex-valued convolution of two complex-valued functions are comparable to the cross correlation between the functions. An example is when a complex filter matrix $W = A + iB$ is convolved with the complex vector z, where A and B are real matrices. The convolution is presented in [20] as

$$W * z = (A * a - B * b) + i(B * a + A * b) \tag{33}$$

which in matrix notation this is represented as

$$\begin{bmatrix} Re(W * z) \\ Im(W * Z) \end{bmatrix} = \begin{bmatrix} A & -B \\ B & A \end{bmatrix} * \begin{bmatrix} a \\ b \end{bmatrix} \tag{34}$$

where $Re(\cdot)$ is the real part and $Im(\cdot)$ is the imaginary part.

The activation function ReLU, as defined in (8) is not compatible with complex numbers. A couple of complex-valued versions are suggested in [20] where the best performing one was found to be the naivest one, called the Complex ReLU, $\mathbb{C}ReLU(\cdot)$ which is defined as

$$\mathbb{C}ReLU(z) = ReLU(Re(z)) + iReLU(Im(z)) \tag{35}$$

which is complex-valued with $a, b \geq 0$, i.e., simply the ReLU for the real and complex part separably.

Batch normalization is a common tool used in training of the neural network. When each of the batches in the training state is normalized, there will be less overfitting and faster training as described in [21]. For the first part of real-valued batch normalization in [21] the mean and standard deviation is calculated from the values in the batch. The mean is then subtracted from the data and finally divided by the standard deviation.

$$\tilde{x} = \frac{x - E(x)}{\sqrt{V(x)}} \tag{36}$$

where x is the real-valued data in the batch. In addition to this normalized data, $\tilde{x}$, a shift parameter β and a scaling parameter μ is trained in the network to recreate normalized values close to the original. The final result in [21] of a batch normalization have the form

$$y = \mu\tilde{x} + \beta \tag{37}$$

where y is real-valued. The standard batch normalization is defined for real values only. To make it compatible for complex values, first the normalization step needs to follow the complex normal distribution. The way the complex-valued batch normalization is defined in [20], is that the normalization part from (36) becomes

$$\hat{z} = \frac{\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}}{\sqrt{V(z)}} \tag{38}$$

where $\begin{bmatrix} \hat{a} \\ \hat{b} \end{bmatrix}$ is a $2 \times n$ vector where the rows correspond to the real and imaginary part of the complex-valued vector $z - E(z)$. The denominator consists of

$$V(z) = \begin{pmatrix} Cov(Re(z), Re(z)) & Cov(Re(z), Im(z)) \\ Cov(Im(z), Re(z)) & Cov(Im(z), Im(z)) \end{pmatrix} \tag{39}$$

which is a 2×2 matrix consisting of real values. Analogously with the real-valued batch normalization, the shift and scaling parameters are trained and $y = \mu\hat{z} + \beta$ is thus formed. Here the scaling parameter is a 2×2-matrix (corresponding to the inverse of the square root of the covariance matrix) formed as

$$\mu = \begin{pmatrix} \mu_{rr} & \mu_{ri} \\ \mu_{ri} & \mu_{ii} \end{pmatrix}. \tag{40}$$

The normalized vector $\hat{z}$ in (38) is centered at 0 but does not follow the complex normal distribution. If the real or imaginary part have different variance, the normalized vector will be elliptic shaped rather than circular, which would have been the case for a complex normal distributed vector.

3.2.5 Postprocessing

In pitch estimation methods such as neural networks, probabilistic or parametric methods, the estimators typically yield approximative probabilities, or something similar, for the candidate pitches to be a fundamental frequency. Most often those probabilities are not simply 1 or 0. In that case the fundamental frequency extraction would be obvious.

In an MLP for a classification purpose, Section 3.2.2, a softmax, (10), was suggested as the final activation function. In order to achieve probability 1 for one of the classes, the activations for all other classes contributing to the softmax must be $-\infty$. This is practically infeasible to achieve by numerical methods, and one must probably trust a system which says that a pitch is significantly active, with probability > 0.95. But one would probably trust lower probabilities as well. To automatically choose which probabilities to accept, the simplest solution is to set a binary threshold. Output pitches with higher probabilities than the threshold are, by the binary threshold method, set to 1 and pitches with lower probabilities are set to 0. The output after the binary threshold has been applied can easily be translated into a music sheet.

The binary threshold can be computed in different ways. One way is to train the threshold by maximizing the MIREX accuracy, (43). By setting a binary threshold one is determining where on the precision-recall scale to be placed. By choosing a high threshold, the precision will be high since only pitches with a high probability will be chosen. By choosing a low threshold, the recall will be high since even the least salient fundamental frequencies will be captured. This can be summarized in the precision-recall curve, Figure 11.

Other versions of postprocessing can also be considered. For example, due to the spectrum envelope of a tone with high power at the onset which then is decaying, the pitches may appear with a higher probability from a neural network or subharmonic summation close to the onset. One may therefore consider lower probabilities close to the offset. Different algorithms for this exists and are explained further in [2].

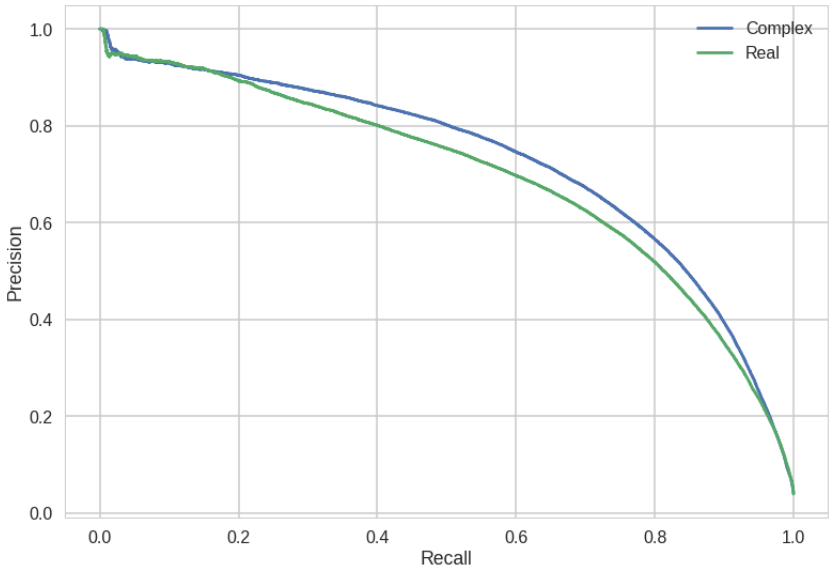

Figure 11: A precision-recall curve of two neural network models called Complex and Real consisting of probabilities $\in [0, 1]$ for each pitch candidate. The curve can be described as a function of the binary threshold. When the threshold is close to 1 the precision is close to 1 and the recall is close to 0, that means that the threshold only let candidates with a high probability to pass. On the other hand, when the threshold is close to 0 the recall is close to 1 and the precision is close to 0 which corresponds to the case where almost all tones are estimated as fundamental frequencies. For optimizing the accuracy, usually the precision and recall should be of similar level which corresponds to a balance of the number of estimates and the probability that they are correct. In this case the Complex curve appears above the Real curve. This yields that the complex-valued model have a higher average precision. The average precision can be interpreted as the area under the curves.

3.3 Available Datasets for Model Training

The main part of the automatic music transcription models is dependent of parameters which in some way are trained. In order to train the parameters, a dataset containing both a music signal and an annotated sheet music is needed, preferable a large amount of such which may represent a broad set of music. Especially the Neural Network based methods are dependent of the amount of training data and would perform poorly if the dataset used for training would be too small. There are problematically not that many datasets available, which inhibits the research. Some of the datasets which are available are listed in Table 2.

Dataset	Instrument	Including	Year
MAPS [22]	Piano	31 GB piano music generated from MIDI files. Consisting of monophonic sounds, random chords, usual chords and pieces of music.	2008
MusicNet [23]	Ensemble & piano	330 classical real music recordings with sheet music annotated by trained musicians.	2016
Bach10 [24]	Ensemble	Real music recordings of woodwind quartet playing 10 different songs by Bach.	2011
RWC [25]	Ensemble	Including 50 pieces classical music and 50 pieces jazz from MIDI files, performed by a variation of ensemble instruments.	2002
MIREX Development set [26]	Ensemble	1 piece of real music recording of a woodwind quintet.	2005
Su Dataset [27]	Ensemble & piano	5 pieces of piano and 5 pieces of string instruments, real music recordings.	2015

Table 2: Available datasets for music transcription.

4 MIREX

MIREX stands for Music Information Retrieval Evaluation eXchange and started up 2000 in a small edition. This is an annual conference arranged by the International Society for Music Information Retrieval (ISMIR) where different researchers from all over the world presents their solutions to different tasks, or estimation problems, in music information retrieval (MIR). The conference, has been growing since the start and for MIREX 2017 totally 22 different tasks where included in the conference, such as Tempo Estimation, Chord Estimation, Classical Composer Identification, KPOP Genre Classification, Cover Song Identification and more [18]. The number of tasks is changing from year to year since new areas of MIR are discovered as time goes by. This follows as the relatively small field of MIR is growing.

One of the tasks in Mirex is called Multiple Fundamental Frequency Estimation and Tracking, which mainly focuses on correctly finding the notes in a music piece. This task was the inspiration to this thesis and it will be in focus. The goal of this task is to identify the active fundamental frequencies in a polyphonic music signal. Compared to other tasks, this one is quite straight forward. Multiple Fundamental Frequency Estimation and Tracking has been a task in the conference since 2007 where it replaced the closely related task Melody Extraction [18].

The task is divided into subtasks, which will be explained in detail in this section. How the subtasks are organized is described in Section 4.1, the design of the test set is declared in Section 4.1.1, how the estimates are measured for Multi-F0 are explained in Section 4.2, and the counterpart for Note Track is described in 4.3.

4.1 MIREX Evaluation

The competing researchers first send in their models a couple of weeks before the conference. For the Multiple Fundamental Frequency Estimation and Tracking task, the models are then evaluated on two similar subtasks which gives different scores. Those two tasks can be read about in [18] and are Multiple F0 Estimation, where each active fundamental frequencies should be presented for each time frame, and Note Tracking where the result is presented as notes with a certain duration, which is more similar to sheet music. Both subtasks are evaluated on two different datasets, the The so called MIREX and Su datasets, introduced 2009 and 2015, respectively, which both are real-world recordings. The MIREX Dataset is kept secret for the competitors while the Su Dataset is open for the public and available through [27][22]. Further in this thesis, when referring to the results, results on the MIREX Dataset are considered if nothing else is claimed[23].

4.1.1 The MIREX Dataset

The MIREX Dataset (used in task Multiple Fundamental Frequency Estimation and Tracking) consists of 40 clips of half minutes long polyphonic music of with 2-5 ensemble instruments [18].

[22]The confidentiality for the MIREX dataset is one of the reasons why the competing researchers needs to send in their models instead of just evaluating and presenting the results themselves.

[23]Because of higher thrust in a secret dataset.

There is a total of 11 songs, each with different combinations of the instruments, resulting in totally 40 music files. Some of the music clips comes from a woodwind quintet transcription of L. van Beethoven's String Quartet Op. 18 No.5[24]. The rest of the clips are RWC MIDI and MIDI samples including pieces from Classical to Jazz collections [18].

For the Note Tracking task, there is a total of 34 music files, where 24 came from polyphonic ensembles of 2-5 instruments and 6 polyphonic piano files. For the Note Track piano subtask, only the 6 piano files are considered, generated using a disklavier playback piano [18].

4.2 Multiple Fundamental Frequency Estimation

The first subtask considered herein is the Multiple Fundamental Frequency Estimation, or Multi-F0 as it is shortened. In the submission format, as described in [18], for each time frame of 0.01 seconds (10 ms), one is supposed to determine which fundamental frequencies are active[25]. The general form is

```
time    F01     F02     F03
time    F01     F02     F03     F04
time    ...     ...     ...     ...
```

and a model output might look like

```
0.78    146.83  220.00  349.23
0.79    349.23  146.83  369.99  220.00
0.80    ...     ...     ...     ...
```

To measure how the different models are performing, the estimates are compared to the ground truth fundamental frequencies. If an estimated pitch is within a quartertone (±3%) from the ground truth it is assumed to be correct[26]. From this comparison one first finds the amounts of true and false positives (TP and FP) and true and false negatives (TN and FN). These measures are understood simply by studying Figure 12.

From these measures one can, as in [29], calculate the two most standard classification measures, precision and recall, as

$$Precision = \frac{TP}{TP + FP} \tag{41}$$

$$Recall = \frac{TP}{TP + FN} \tag{42}$$

and in text, precision describes how probable an estimated pitch was to be correctly estimated, while recall describes how probable a true pitch is estimated active. Both measures are bounded by $[0, 1]$ where a score close to 1 is considered good and a score close to 0 is considered bad.

Precision and recall are in some sense describing how good an estimate is. But, for example, if one would estimate only one pitch to be active, and it turned out to be correct, the precision

[24]Even if the song is known, the woodwind quintet transcription is not known. Also, this specific music piece is about 25 min, so it is not obvious which parts are included in the dataset.

[25]ms stands for Milliseconds and the fundamental frequencies are measured in Hz.

[26]In the evaluation of an estimate in MIREX, only one true fundamental frequency can be associated with one estimated pitch. That means, for example, that if two true pitches differ by a quarter tone and a pitch is estimated in between them, it can only be associated with one of the true pitches.

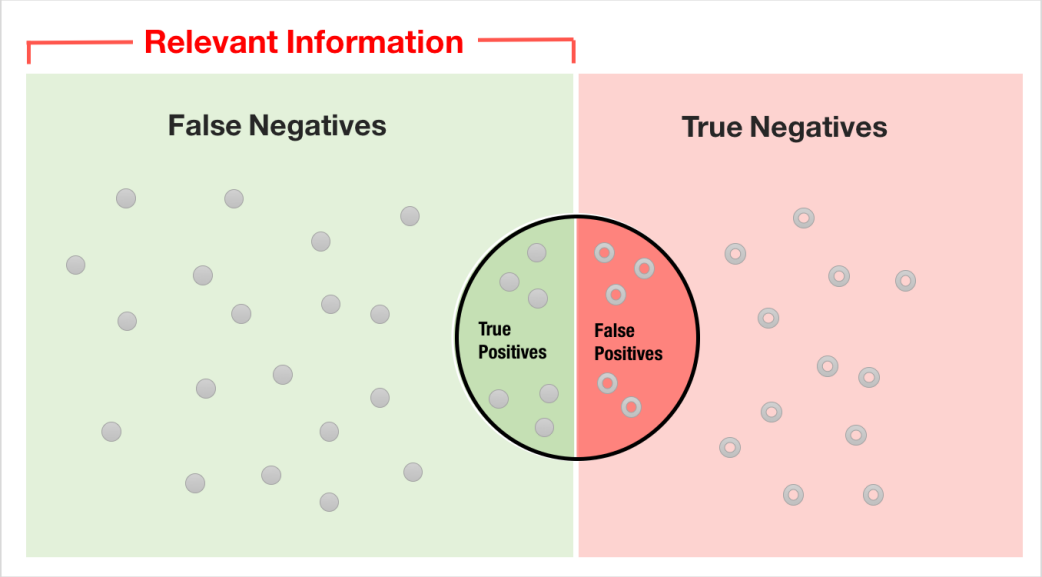

Figure 12: The green area represents the ground truth pitches and the red area represent the pitches which are not active. The area within the circle represents the estimated pitches and the area outside the circle represents the pitches which are estimated to be not active. True positive is the green area inside the circle, that is the amount of correctly estimated active pitches. False negative is the green area outside the circle which is the amount of active pitches which were mistakenly estimated to be not active. False positive is the red area inside the circle and corresponds to the pitches which are mistakenly estimated to be active while they were not active. Finally, the true negative is the red area outside the circle which corresponds to the pitches which were correctly estimated to be not active. The image is presented in [28].

score would be 1 even if the other pitches were not found. And conversely, if one estimates all possible pitches to be active all the time, one would get a recall score of 1, even if most of the estimated pitches are truly inactive.

Therefore, another measure is often used as the primal measure when evaluating models, termed Accuracy. In MIREX, the accuracy is defined as [29]

$$Accuracy = \frac{TP}{TP + FP + FN} \tag{43}$$

and one should note that the MIREX accuracy differs from the formal derivation of accuracy as [30]

$$Accuracy^{(*)} = \frac{TP + TN}{TP + FP + TN + FN} \tag{44}$$

where the TN term is added. If the TN would have been included in the MIREX accuracy in (43), one would achieve a high accuracy score by simply estimating no active pitches at all, which is why the TN term is excluded in the MIREX version of accuracy[27].

Precision, recall, and accuracy are the three measures this thesis will focus on while evaluating different Multi F0-models. In MIREX, a couple of other additional measures are evaluated, often

[27]If for example 4 pitches out of 84 would be active in the ground truth and 0 pitches were estimated by the model, TN=80. This would give the accuracy a score of $\frac{80}{80+4} \approx 0.95$. This high accuracy score wouldn't give a fair score of the model.

termed error scores. These are used mainly to describe what type of errors that appear in the estimates from the model. A perfect estimation will achieve an error score of 0. The error scores mentioned in this section, are however, not used to analyze the models considered herein. First, E_{Sub} is defined as [31]

$$E_{subs} = \frac{\sum_t min(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_t N_{ref}(t)}$$

where $N_{ref}(t) = TP(t) + FN(t)$, which is the number of true pitches in frame t. $N_{sys}(t) = TP(t) + FP(t)$ is the number of estimated pitches at frame t. $N_{corr}(t) = TP(t)$ is the correctly estimated pitches at frame t. E_{subs} gives a measure similar to recall since it counts how many of the ground truth pitches which are actually estimated, but it is less robust than recall since an underestimating model (which estimates less pitches than the ground truth) can also get a good score. Actually one will get a perfect score of 0 if the model either estimates all pitches, no pitches at all or exactly the correct pitches. The measure is bounded by $[0, 1]$ where a score close to 0 is sought. The second error score is E_{miss} which is defined as [31]

$$E_{miss} = \frac{\sum_t max(0, N_{ref}(t) - N_{sys}(t))}{\sum_t N_{ref}(t)}$$

which is indicating if the model estimates too few pitches. If a model estimates exactly the same number of pitches or more it will achieve an error score of 0. The measure is bounded by $[0, 1]$. The E_{fa} is defined as [31]

$$E_{fa} = \frac{\sum_t max(0, N_{sys}(t) - N_{ref}(t))}{\sum_t N_{ref}(t)}$$

and is the opposite of E_{miss}. E_{fa} is a measure which indicates if a model estimates too many pitches. If the model estimates exactly the same amount of pitches or less, it will get a good score close to 0. The measure is non-negative but not upwards bounded. The final error score E_{tot} is defined by [31]

$$E_{tot} = \frac{\sum_t max(N_{ref}(t), N_{sys}(t)) - N_{corr}(t)}{\sum_t N_{ref}(t)}$$

which is a shortened formula of the sum of all the other error scores together. One finds this measure to be the only robust of these error scores, since only the perfect model will get a score of 0. This score is non-negative but not upwards bounded.

For the Mulit-F0 task the 7 measures (precision, recall, accuracy, E_{tot}, E_{subs}, E_{miss}, E_{fa}) are also evaluated for chroma results. This means that all estimated frequencies are mapped to a single octave before the evaluation is done. The reason why this is done is since one of the most common mistakes when estimating the fundamental frequency, f_k, is to mistakenly estimate the half tone $\frac{1}{2}f_k$ (also called the sub octave) or the first overtone $2f_k$[28]. In some areas one might only be interested in the chroma tones. That gives two reasons why this transcription is examined in MIREX. Historically in MIREX, one finds that the rankings in the chroma list of result don't differ that much from the rankings in the standard list of result [18].

The final MIREX performing measure presented in the Multi-F0 task is the Friedman test, where differences among the models' performances are examined as significant or not. The performance

[28]See Section 3.1.2 and 3.1.3 for details about overtones.

is measured in accuracy, being considered as the most overall measure. The Friedman test mainly consists of a Tukey-Kramer HSD Multi-Comparison [32]. Pairwise, student T distributed-based, confidence bounds between each model's accuracy measure are formed by

$$I_{i,j} = \mu_i - \mu_j \pm \frac{q_{\alpha,df}}{\sqrt{2}} \hat{\sigma}_\epsilon \sqrt{\frac{2}{n}}$$

Where μ_k is the accuracy for model k, q is the student T value for the confidence rate α (typically 0.95) with degrees of freedom df. The estimated standard deviation $\hat{\sigma}_\epsilon$ is calculated from all models (not just the two that are compared), and n is the number of test samples. If 0 is in the interval $I_{i,j}$, then the model i and model j cannot be proven to perform statistically different [32]. The spread of each model performance is considered by the test, which can be claimed to be the most correct way of comparing models. However, in the work of this thesis, the Friedman test would be too complicated to apply because of its parameters being determined from all involved models. The test will thus not be considered further in this thesis.

4.3 Note Tracking

The second subtask considered herein is Note Tracking. This format is more closely related to the way a person reads music from a music sheet. There are different ways to symbolically represent a tone in a music sheet, but here the piano-roll convention is used. The piano-roll submission format in [18] for each estimated tone, consists of: An onset time, an offset time and a single estimated frequency. That is when the tone starts, when the tone ends and what pitch. The general format [18], looks like

```
onset    offset  F01
onset    offset  F02
...      ...  ...
```

which may look like

```
0.68     1.20     349.23
0.72     1.02     220.00
...      ...      ...
```

which appears easier to translate to a music sheet than the Multi-F0 format. To classify a tone to be correctly estimated, the estimated pitch first of all have to be within a quartertone (±3%) from the reference pitch. Also the estimated onset needs to be within ±50 ms from the reference onset. Finally the offset needs to be within 20% range from the true tone offset (the percentage is depending on the length of the true tone)[18]. All these three conditions need to be fulfilled in order to classify an estimated tone as correct. Similarly as for the Multi-F0 task, only one ground truth tone can be associated with one estimated tone. For Note Tracking the measures precision and recall are calculated in the same way as for Multi-F0 in (41) and (42). Here one may note that the number of correct and incorrect estimated tones, generally is less than the correct and incorrect estimated frequencies each time bin for Multi-F0, making the precision and recall for Note Tracking more sensitive.

In Multi-F0, accuracy was used as the main measure. For Note Track, this is replaced by the F-measure. The F-measure is defined in [29] from the precision and recall by

$$F = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

This measure is bounded by $[0, 1]$ and gives a fair average of the precision and recall. It is robust against shortcuts such as maximizing only one of the precision or recall[29]. In order to maximize the F-measure, the precision and recall should in general be of similar levels.

There is a fourth measure (in addition to precision, recall and F-measure) for Note Tracking called Average Overlap. This measure will only be mentioned here but not considered any further in this thesis. The average overlap measures, for each ground truth tone, the ratio of how well the onset and offset matches the closest estimated tone's onset and offset. The ratio is determined by combining the reference and the estimated onset and offset. The Overlap Ratio is, as in [29], defined as

$$OR = \frac{min(ref_{offset}, est_{offset}) - max(ref_{onset}, est_{onset})}{max(ref_{offset}, est_{offset}) - min(ref_{onset}, est_{onset})}$$

where the inner interval of onset to offset is divided by the outer interval of onset to offset[30]. To achieve a high Average Overlap score, the estimated onset and offset should be as identical as the reference onset and offset as possible. The Overlap Ratio is bounded by $[0, 1]$.

For the Note Tracking task, these 4 measures are evaluated for chroma results in the same way as for Multi-F0 (Section 4.2). This will, in the same way as for Multi-F0, not be considered further in this thesis. In the Note Track task in MIREX one test is on ensemble music and one test is on piano music [18]. This is taken into consideration further when designing the test dataset, described in Section 6.1.

There are also results presented without considering the offset. Thus an estimated tone only needs to fulfill the criterion of being within the quartertone and to have an onset within ± 50 ms. This is of interest because the definition of the offset definition is quite vague, Section 3.1.4, and in addition it is often the most crucial criterion. To illustrate the vague definition of an offset one may again consider a Figure 1. Because of the shape of the signal it is not so easy to determine exactly where the tone ends and where the offset should appear. The way it is determined practically for the ground truth, is that a threshold is set and when the amplitude of the tone is below this threshold, the tone is considered as inactive. From Figure 1 one finds a peak when the tone starts, making it quite easy to identify where the amplitude will be above the threshold and where the onset should appear. To find where the amplitude will fade below the threshold is not as obvious and therefore the offset is more complicated to determine.

Finding when a tone starts and what pitch the tone has may be considered being the most important information. Along with the knowledge that those two criteria are the most trustworthy in the ground truth, one may in some cases consider the Note Track representation without the offset to be the fairest one. Further in this thesis, both the standard result including the offset criterion and the result excluding the offset criterion will be considered.

[29] Ways of maximizing precision and recall one by one was mentioned in Section 4.2.

[30] In MIREX there are some low-performing models which achieves negative average overlap which contradicts this definition of the overlap ratio. Thus the MIREX version of the average overlap and the one in [29] may differ. Anyway, the MIREX version of overlap ratio is not expected to differ that much from this one.

In the same way as for Multi-F0, a Friedman test is evaluated for the models, using Tukey-Kramer HSD Multi-Comparison. Here the significance tests are based on th F-measure which is considered to be the most general measure. However, this measure is deemed to be of limited interest in this thesis and will not be considered in the evaluations.

5 Models of Multiple Fundamental Frequency Estimation and Tracking

A main part of this thesis consisted of understanding and evaluating the most recent methods used in the research area of polyphonic music transcription. This was done by finding interesting models, understanding how the models should be executed and then evaluating them on a created test set. Afterwards the models have been analyzed in order to explain different behaviours in the test results.

In this section, first a description of which models are considered in this thesis, Section 5.1, then each of the considered models in MIREX will be introduced and shortly explained together with a summary of their result in MIREX along with technical details, Section 5.2-5.4, followed by a summary of MIREX 2015-2017 in Section 5.5, then a similar introduction to models which haven't participated in MIREX in Section 5.6. Then two herein proposed models (which are variants of the model in Section 5.6.4) are presented in Section 5.7, and finally in section 5.8 the models are summarized and compared, in order to identify some trends of the models. All results from MIREX which the following sections will refer to, are found in [18].

5.1 Evaluated Models

Ever since the late 90's, many different models has been suggested considering polyphonic music transcription. To consider them all would not be possible in the scope of this thesis. A selection of models was therefore considered mainly based on models published in conjunction with MIREX conference. While aiming to get the most recent research, the models published in MIREX 2015-2017 were considered. All models aimed at solving the fundamental frequency estimation problem are, however, not published in conjunction with MIREX and some effort was done to track these. Such methods considered herein were found from suggestions by researchers in the area.

The documentation one gets from the participating models in MIREX is a short report, referred as an abstract, for each submitted model. To be able to evaluate the models, the corresponding authors were contacted through e-mail and asked to share their code of their model[31]. Most of participants sent code for their trained models while some returned their model estimates by e-mail after receiving the test files[32]. The models which weren't reached from MIREX was either available on the Internet or delivered through e-mail.

Some participants submitted more than one model. Examples of this could be one version for Multi-F0 and one for Note Track or one model trained on ensemble music and one model trained on piano music. A third example is someone sending both a slow model, as well as a fast but less accurate model. All these versions will not be considered in the thesis, instead a selection

[31]Not all models from MIREX 2017 and 2016 are presented in this thesis. Time limited forced a selection of models to be considered, thus 6 models were disregarded. The disregarded models were: PRGR1, PRGR2, WCS1, ZCY2, SL1 (MIREX 2017) and KB1 (MIREX 2016). The best performing models in MIREX 2017 which were not considered in this thesis was WCS1 which came in 5th place in the Multi-F0 task and PRGR1 which came in 3rd place in the Note Track task. For MIREX 2016 KB1 came in 5th place in Note Track (all positions are based on the accuracy or F-measure depending on the Multi-F0 or Note Tracking task). All the disregarded models were in MIREX stated to be significantly lower performing than the best performing model the corresponding year of participation. Thus, this thesis can be claimed consider the best performing models from MIREX.

[32]This was done for the authors who, for publishing reasons, could not send their complete models.

will be made, motivated in each specific subsection.

The models considered, do in general have a long name, referring to how it works. To keep notation simple, a long name for a model has been changed into the initials of its author. If there were more than one author, the first letter in the surname of each author is used. If multiple models by the same authors are considered, the name of each model will be appended by different numbers.

5.2 MIREX 2015 Participating Models

MIREX 2015 had a quite thin lineup, with only 3 participating research teams (compared to MIREX 2014 which had 9 research teams). However, each team instead contributed with 2-4 models, which resulted in a total of 9 participating models. In short, for the Multi-F0 task, the clear winner according to the accuracy measure was model BW1 followed by model MPE1. For Note Tracking model BW2 performed the best on the mixed music dataset followed by BW3. On the subtask with piano music there was reversed rolls where the BW3 came 1st followed by BW2. These will now be briefly presented.

5.2.1 BW

Overall model BW can be seen as the winner of MIREX 2015, both in the Multi-F0 and the Note Track task. The model for MIREX 2015 was an improved version of their model 2014, which ended up in 3rd place for both tasks.

The BW model for MIREX 2015 consists of three similar versions [33]. BW1, which is trained on ensemble music for Multi-F0, BW2, which is trained on ensemble and piano music for Note Tracking, and finally BW3, which is trained on piano music only for Note Tracking.

The model applies the Equivalent Rectangular Bandwidth to the input data, see (3). It is claimed that the ERB gives a better temporal resolution while simultaneously keeping the frequency axis compact [33]. The model then uses the probabilistic latent component analysis (PLCA), where the probability of a certain tone at a certain time step is calculated based on the sound state, pitch and instrument [33]. This can be understood clearer by considering a tone played by a piano. A piano tone has different evolutions through a tone, referred as attack, sustain, and decay. Also it is probable that a tone in the middle register (somewhere between 110-880 Hz) is played. These unknown probabilities are updated for each time step until they converge, using the expectation-maximization (EM)-algorithm. These weighted probabilities are used when calculating the probabilities of a certain pitch at a certain time step. If the probability is higher than a set threshold, the pitch is considered as active [33].

The model was written by Emmanouil Benetos and Tillman Weyde at Queen Mary University. The code of the model was written in Matlab. The training data came from the RWC database [25] for all instrument except piano where the MAPS dataset [22] was used.

5.2.2 CB/Silvet

CB, or Silvet as it is referred to in the literature, is one of the most recurrent models in the MIREX competition. This model submission is identical to their model submitted 2012, as well

as every year since. In MIREX 2015 it is performing averagely. Their top placement in MIREX 2015 occurred at the Note Tracking for piano music task with a 2nd place. At Note Tracking for ensemble music, the model came in 3rd place.

The model is superficially described in [34] and more in detail in [35] and is mainly created for Note Tracking, but the authors have also converted the model estimates to the Multi-F0 format. They also train the model on ensemble music and piano music separately, which equates a total of 4 models. In this thesis, the Multi-F0 model for ensemble and piano music is analyzed as well as the Note Track model for ensemble music (Note Track for piano music is disregarded). There also exists a so-called Silvet-live model, which is essentially faster than the default Silvet at the cost of results. Since the running speed of the models are not directly measured in this evaluation, this model is not considered in the thesis.

Silvet is a multi-feature model which includes a couple of different music feature extractions such as onset/offset detection, beat/tempo estimation, instrument classification and key/chord detection. Silvet uses these features together with CQT-transformed data for estimating tones in Note Tracking using a probabilistic latent-variable estimation method, similar to the earlier described model BW (Section 5.2.1) [34]. In the model a couple of thresholds are trained from the mentioned features which decides the note activations. The training is done on solo instrument recordings and source separation is performed on the input data in order to estimate the different parts of the signal [35].

The Silvet model is created by a couple of researchers at Queen Mary, University of London. The researchers which are referred as the authors of the music transcription part of the model are Chris Cannam and Emmanouil Benetos. [33]. The code of the model is written in $C++$. The training data is from the MAPS dataset [22] and MIDI files extraction from a Disklavier piano. Data from the RWC [25] database and the MIREX development set [26] are used for training for the ensemble parts (which is criticized by the authors to be too small datasets).

5.2.3 SY/MPE

The SY model is in the literature also referred to as MPE. In MIREX 2015, 4 variants of the MPE model participated, where MPE1 was the one performing the best in both Multi-F0 and Note Tracking. MPE1 achieved the 2nd place both in Multi-F0 and in Note Tracking. It isn't very usual that a single model performs on such high level in both tasks. It is worth to mention that the MPE1 achieved 1st place in Multi-F0 for the Su dataset, but some criticism to these results are motivated since the authors of MPE are the same as those who created the Su dataset.

The model is considered robust due to its high performance in both the Multi-F0 and the Note Tracking. Concerning what differentiates the 4 different models, the available abstract [36] doesn't indicate how. A guess is that they are either trained on different datasets or have different thresholds. In order to be prudent, all 4 models were considered in this thesis.

The model's authors also propose a new approach to music transcription called combined frequency and periodicity. The model is based on the theoretical framework for single pitch estimation [37]. The spectral and a temporal representation is calculated both in form of a log-scaled

[33]It may be noted that Emmanouil Benetos also was involved in model BW in Section 5.2.1.

spectrum and the logarithm cepstrum. The log scaled spectrum is the logarithm of the absolute value of the DFT or similar. The power cepstrum is

$$|\mathcal{F}^{-1}\{log(|\mathcal{F}\{x_t\}|^2)\}|^2$$

where $\mathcal{F}$ is the Fourier transform and $\mathcal{F}^{-1}$ is the inverse Fourier transform. The formula describes the squared inverse Fourier transform of the logarithm of the squared Fourier transform. The authors claim that the cepstrum directly yields the subharmonics or the fundamental frequency, and is robust against so called octave errors, i.e., nf_0 for all $n \in \mathbb{Z}$, where the fundamental frequency is denoted f_0. The spectrum of a pitch signal consists of the fundamental frequency and its overtones, and [36] claims it to be robust against sub-octave errors, i.e., $\frac{1}{n}f_0$ for all $n \in \mathbb{Z}$. Combining the spectrum and the cepstrum in a smart way should thus be robust against both sub-octave errors and octave errors. This assumption is applied to a multipitch scenario and a pitch estimation function is applied[34].

The MPE models are created by Li Su and Yi-Hsuan Yang at Research Center for Information Technology Innovation, Academia Sinica , Taipei, Taiwan. The code was written in Matlab. The training data is unknown from the abstract, but it is not unrealistic to assume that parts of the Su Dataset was used, as it was also created by these authors.

5.3 MIREX 2016 Participating Models

MIREX 2016 had, just like the previous year, quite low participation. The competition was between 4 participating research teams where the number of participating models was 5. In short, for the Multi-F0 task the model MM1 outperformed the other ones while the model DT1 was the best performing model on the Note Track task both on the mixed dataset and the piano dataset, based on the F-measure [18]. The model KB1 was participating but was not considered in this thesis due to its bad score[35].

5.3.1 CB/Silvet

CB is essentially the same model as described in Section 5.2.2. It performs moderately well and achieves a 2nd place in the Multi-F0 task at MIREX 2016 and a 3rd place in Note Tracking for both subtasks. Recalling MIREX 2015, Silvet was performing better in the Note Track tasks than the Multi-F0 task, which suggests that Silvet is quite universal.

5.3.2 DT

The model DT achieved an impressive score on the Note Track task, resulting in a 1st place both for the ensemble and the piano music subtasks. On the Multi-F0 task it was placed 3rd.

The DT model described in [38], is one of many participating models this year which are based on neural networks. This specific model uses CNN, as described in Section 3.2.3. The model uses a variant of the CQT where non-linear scaling is added together with some form of normalization. There was a whole number of frequency bins per note for a pooling layer to not mistakenly merge frequencies of two different tones. The model then detects where onsets may appear in

[34]Further details about the model, such as exactly how the spectral and temporal representation should be designed or how the pitches are declared, are not available through the given available abstract [36].

[35]Which was partly due to bugs in their submitted code.

the signal. This is done by considering a spectrogram and identify where many frequency bins increased in power in a short period of time. The spectrogram is then divided into rectangular windows from one onset to the next. These sliced spectrograms were then used as input to the CNN [38].

The CNN architecture is based on a state-of-the-art image detection architecture [39]. The output layer in the CNN is an array of length 88 where each neuron represents a tone [38]. A difference from the CNN described in Section 3.2.3 is that the sigmoid (7), was used as activation function. This results in approximative probabilities (or independent probabilities)[36]. In addition to these probabilities, some filtering was made in order to detect separate notes. All neurons which received a probability over a certain threshold were considered as fundamental frequency candidates, and the rest were set to 0. The probabilities appear as peaks at different frequencies which were ordered from most probable to least. From the empirical results in [38], one of the fundamental frequencies in the signal is typically likely to have the strongest peak. This peak is then selected to be considered as a fundamental frequency. All candidate frequencies of a multiple of the selected frequency is disregarded of consideration of being a fundamental frequency. The next fundamental frequency is determined from the rest of the candidates, and so on, until no candidates were left. Finally the CNN output probabilities and the filter candidates are combined to produce a final estimate. For each estimated fundamental frequency, the onset was estimated using onset detection and the offset was set to the time point where the next onset was estimated [38].

The DT model was created by Daylin Troxel at the company Lunaverus. The network was trained to maximize the standard accuracy (44). The model was trained on 2.5 million of samples auto-generated from 3000 MIDI files. The samples in MIDI-format were translated to .wav-format. The code for the model was not open source. In order to receive the test results from DT1, the test files were sent by e-mail to the author who then returned the estimates.

5.3.3 MM

The MM model performed very well at MIREX 2016. In the Multi-F0 task, it came on 1st place, and on both subtasks in Note Track it was placed 2nd.

The MM model stands out from the other models in a special way, even if it is one of many submissions this year which are using neural networks. The model is submitted in MIREX 2016 but the article which it is referred to is published 2004. The model's article [40] states that no previous references to works that uses neural networks for music transcription was found. The MM model was way ahead of the research back then and it was a pity that this work didn't get more attention in order to inspire the research area to start elaborating more with neural network for music transcription.

From its paper, [40], it is quite clear that it is a couple of years old. Many tools which are used in this model has not been seen in the other music transcription models in more recent years. Many concepts in this work needs some explanation and a lot of theoretical aspects should be added to understand this model completely. This extra work is considered unnecessary for the scope of this thesis, therefore only some parts of this model are explained.

[36] Which sum could become above 1.

The model consists of the two main parts. In the first part the music signal is presented as a spectrogram on the time-frequency scale and the second part is where the notes are recognized. Both parts are using neural networks in different ways. The first part divids the signal into frequency channels by so called gammatone filters and a hair cell model. In the second part, where a spectrogram is given from the frequency channels, a set of 76 different neural networks are introduced. Each network's mission is to determine if a certain frequency is in the signal. The use of 76 networks and not 88 (which is the amount of tones on a piano, Section 3.1.2), is motivated by the claim that the 10 ms of data considered in each spectrum makes it impossible to distinguish low frequencies. Therefore the lowest octave is disregarded[37]. This is a bit different from the described neural network for music transcription in Section 3.2.2. In this case the final layer consists of a single neuron where a high value indicates that the corresponding frequency is in the signal and a low value that it is not.

It should be mentioned that this model is quite complicated in the sense that many features are added to the model. To determine the active frequencies by the 76 networks, first a partial tracker module is used in order to reduce octave errors [40]. Also an onset detector using the MLP is added to the networks. The neural network model which is used is a time-delay neural network which often is applied in speech recognition. Given the estimated active frequencies, a detector of repeated notes is added to the model. This is supposed to determine whether a pitch which is active for a long time sequence where multiple onsets are appearing, is one long tone or multiple repeated shorter tones. This detector consists of an MLP. Finally, the length and loudness of the tones are estimated, and the transcription is complete [40].

The MM model is created by Matja Marolt at University of Ljubljana. The model was trained on a dataset consisting of a set of 120 MIDI piano pieces of various styles including classical music, ragtime, jazz, blues and pop. To make the dataset able to recognize low and high tones which didn't occur in the MIDI dataset, synthesized piano pieces with one to six tones played simultaneously. Totally the final dataset contains 300'000 pairs of input-output patterns. Each of the networks were trained on a smaller amount where 1/3 of the datasets included its corresponding frequency. The code was written in C++.

5.4 MIREX Participating Models 2017

MIREX 2017 had a total of 10 participating research teams with a total of 14 participating models. The high number of participants increased the quality of the competition. In the Multi-F0 task model THK1 performed best, followed by KD2 on a second place, and MHMTM1 on a third place[38]. In the Note Track task for ensemble music model, CT1 achieved the 1st place, KD2 came 2nd and CB1 came on third. In the Note Track task for piano music CT1 still ended 1st, CB1 came 2nd and PR1 ended 3rd.

5.4.1 CB/Silvet

The Silvet (CB) model is familiar from Section 5.2.2 and 5.3.1. The model is perhaps not the best, but it is performing fairly good every year. This year the performance in Note Track was

[37]It is also mentioned in [40] that so low tones are very rare in general music.

[38]KD1 got exactly equal score as KD2 on the Multi-F0 task but no result at all at the Note Track task, therefore it is disregarded.

the best, and CB came on 3rd place in ensemble music and a 2nd place in piano music, which is impressive.

5.4.2 CT

The model CT only participated in the Note Tracking task, in which it performed best of all models in both subtasks. In the piano subtask the superiority was clear while the ensemble music was not significantly better than the second placed model.

The CT model, as described in [41], is built on CNN but includes recurrent layers. As an input to the network, the data is preprocessed by a variant of the CQT called sliCQ. The resulting spectrograms are log-scaled power spectrograms and finally stacked in the depth dimension[39]. The spectrograms are then used as input. There are two separate channels introduced, one called articulation and one sustain. The sustain channel is 1 whenever some kind of sound appears and 0 whenever it is quiet. The articulation channel is 1 at the specific points where a note event occurs, i.e., when an onset appears. The neural network performs a succession of convolutions and average pooling operations. They are followed by a LSTM layer with convolutions, called ConvLSTM. Layer normalizations are also applied. The final layer is a network-in-network layer, which is a variant of deep convolutional neural network. A sigmoid activation, i.e., (7), is applied in the final layer. Finally the postprocessing consists of combining the estimated active frequencies with the articulation and sustain channels to identify the onset and offset of each tone [41].

The CT model is created by Carl Thomé and Sven Ahlbäck at DoReMIR Music Research AB. The dataset by which the model is trained is not specified in the abstract [41], neither is the source code language. Similarly to the DT model in Section 5.3.2, the code was not open source and in a similar fashion, estimates on the test data were received through e-mail.

5.4.3 KD

The KD model performed very well in MIREX 2017. In the Multi-F0 task it came 2nd just as it did in the Note Track task. For the Note Track piano subtask it did not perform as well, achieving a 5th place.

KD is described in [2] and is a parametric pitch estimation model. A multi-resolution FFT is used as input, implemented by the model author in [2]. The multi-resolution FFT is the FFT with adapting frequency and time resolution. When applying the standard FFT there is always a resolution issue whether one wants high frequency resolution or high time resolution. Since high frequencies have higher resolution with regards to semitones, the loss of frequency resolution is not very critical[40]. Therefore higher frequencies can be estimated with a higher time resolution. On the other hand, lower frequencies need higher frequency resolution, so the loss of time resolution is argued being a price worth paying. If one reasons that lower octaves are usually used to play chords, the time resolution perhaps isn't that essential. The multi-resolution FFT adapts the time-frequency resolution according to the frequency scale in the described way [2].

From the resulting spectrogram of the multi-resolution FFT, subharmonic summation is applied for pitch estimation, described in Section 3.2.1. For tones played by instruments, there is usually

[39]The power spectrogram denotes the squared amplitude spectrogram.

[40]Higher frequencies have higher resolution due to the logarithmic behaviour of sound and music.

a strong emphasis in the beginning of the tone followed by a slow decline in the power. This expected envelope is used to predict the onset and the offset of the tone in [2]. The envelope of the tone is estimated in combination with an onset detection for each pitch. The pitch candidates from the subharmonic summation are then used in an iterative process where first the pitch which is most probably a fundamental frequency is considered to be extended from the signal, as described in [2]. Some further analysis is then made whereafter the second most probable pitch is considered and so on.

The KD model is created by Karin Dressler at Technische Universität Ilmenau, Germany. The model algorithm consists of many combined individual steps where parameters for each step are trained on the MIREX development dataset from 2005 [26], less than 1 minute of audio. Nevertheless it is the most similar dataset to the test data which MIREX will evaluate the model on, making it a good approach in order to get a high score in MIREX. The code of the model was private, but the estimates were achieved by an executable file.

5.4.4 MHMTM

The MHMTM model consists of two submissions which perform quite similarly, but MHMTM1 was slightly better and achieved the 3rd place in the Multi-F0 task. It was not participating in the Note Track task.

The MHMTM model in [42] is based on a CNN. The difference between the two submitted models is that MHMTM1 is trained on all types of music such as solo piano, piano together with other instruments and ensemble music, while MHMTM2 similarly is trained on different types of music, but an estimator is categorizing which type of music the audio consists of, for which the appropriate network is used. When the code of the model was achieved from its author, only the code for MHMTM1 was attached, so that is the only MHMTM model which will be evaluated herein.

In this model, the audio first is preprocessed by the CQT, forming the spectrogram matrix. The time step between each centralization of the spectrums in the spectrogram is 0.02 seconds and different overlap ratios between the spectrums were applied. The frequency axis reaches from the highest to the lowest tone on a piano, and each octave is divided into 36 frequency bins. The spectrum is then normalized and the logarithm of the spectrum calculated [42]. The CNN network is based on the same CNN image detection architecture, see [39], as the DT model in Section 5.3.2. The input corresponds to 0.1 seconds of data, corresponding to 5 spectrums, which were stacked, and the output is an array of 88 neurons corresponding to a probability for each pitch[41]. A binary threshold is set using the first approach in Section 3.2.5.

The MHMTM model was created by a research team consisting of Shinjiro Mita, Gaku Hatanaka, Alexis Meneses, Nattapong Thammasan and Daiki Miura at Osaka University. The source code was written in Tensorflow. The training data consisted of a selection of 130'000 MIDI files consisting of pop, rock, jazz, classical music and videogame music. The MIDI files were transformed by custom programs for the piano and ensemble music to sound more realistic.

[41]The stacking can be compared to an RGB image which has 3 color dimensions which are stacked, Section 3.2.3.

5.4.5 PR

The PR model was performing with mixed results in MIREX 2017. In the Multi-F0 task it came 10th, on the Note Track it came 5th for the ensemble music, and 3rd for the piano music.

Only one version of PR (PR1) was submitted to MIREX, but in this evaluation both a PR1 and a PR2 model are evaluated. The reason is that the model consists of two separate versions, one for Multi-F0 and one for Note Track outputs. In this evaluation both outputs are calculated, and afterwards translated to the other format (this will be explained further in Section 6.2), such that both models will be evaluated in both tasks. In this evaluation PR1 is originally given on Multi-F0 format and PR2 originally on Note Track format.

The PR model is a technically advanced model [43]. The model consists of one signal processing part and one pitch tracking part. The authors propose a signal processing technique called Dynamic Mode Decomposition, which consists of transforming time segments of audio data to a spectrum form similar to the FFT. The technique is based on eigenvalue decompositions of linear autonomous systems and is considered outside the thesis' scope. The peaks in the spectrum are then identified and the salience of each pitch is calculated by taking both the spectrum, the timing and the harmonics into account. The pitches are then sorted by the salience, if the pitch with the highest salience is above a set threshold it is estimated as a fundamental frequency. All other pitches which appear as overtones are penalized, after which the second highest salience is considered, and so on.

The PR model was created by Leonid Pogorelyuk and Clarence Rowley at Princeton University, USA. The source code was written in Python. The data on which the model was trained was not specified in the abstract [43].

5.4.6 THK

The THK model only participates in the Multi-F0 task where it performed very well and came on 1st place by margin.

The THK model is briefly described in [44], and in more detail in [45]. The model is built on CNN, similarly to many other models. But there are a couple of things in the model which stand out. First of all the audio data is transformed using a hand-crafted filter bank consisting of 512 cosine-windowed filters. The audio data is transformed to a frequency representation with logarithmic spaced frequency bins from 50 Hz to 6000Hz [44]. The handcrafted filter bank is motivated by gained advantages in a CNN. Each filter considers 1/12 seconds of audio. The stride of the filters is 10 ms, and stacking 25 filters results in a spectrum of 25 dimensions covering about a 1/3 second of audio which is used as the input for the neural network [44].

The network architecture is interesting because it is not very deep, with only two hidden layers. The first layer is a convolutional layer using 128 hidden nodes with a stride of 2. The second hidden layer is a fully connected layer with 4096 hidden multidimensional nodes. The resulting output is matrix of size 193×4096 [44]. This output is fed to a linear classifier for each pitch. This classifier is a linear regression model which is trained in order to minimize the square loss. If the classifier for a certain pitch is over a set threshold it is considered to be a fundamental frequency [45].

The THK model was created by John Thickstun, Zaid Harchaoui and Sham M. Kakade at University of Washington, USA, together with Dean Foster at Amazon. The model is trained on the MusicNet dataset [23] which is a dataset containing both ensemble music and piano music constructed by the same authors. To reduce overfitting, the audio was randomly stretched by up to 5 semitones when the model was trained. Also the speed was randomly raised and decreased. By changing the training data, each training sample would not appear exactly similarly twice and the network would thus not overfit the model [44]. The code was written in Tensorflow.

5.5 Result from MIREX 2015-2017

In this section the results from MIREX 2015-2017 is summarized in Figure 13a, 13b and 14. The tables are extracted from the MIREX summarized evaluation results for each year [18]. The results for both the MIREX Dataset and the Su Dataset are presented. The measure which is considered for the Multi-F0 task is accuracy. For the two subtasks of Note Track, the result without the offset criterion is presented with both the average F-measure and the average overlap, and done so for both the MIREX Dataset and the Su dataset.

Among the results in MIREX 2016 and 2017 in Figure 13b and 14, the top row in each column represents the overall best score in MIREX for the corresponding measure. This gives a guideline of what the state-of-the-art can perform, which is what this thesis is supposed to answer. The use of this thesis can however be motivated by its content of thoroughgoing analysis of why certain models get higher scores than others.

By studying Figure 14, the overall best scores are stated for all MIREX years. Apparently the best Multi-F0 accuracy score was achieved 2014 and is therefore not presented in this thesis, which is unfortunate. But the THK1 model is only slightly behind with a score of 0.003 less than the model from 2014. For the Note Track mixed dataset, a model from 2014 has the highest F-measure, and the difference to the best model in 2017, CT1, is more significant. For the Note Track piano subtask, the best score was achieved 2016 by DT1, slightly better than CT1.

Multi-F0 Estimation

SubID	Participants	Accuracy	
		MIREX Dataset	Su Dataset
BW1	Emmanouil Benetos, Tillman Weyde	0.654	0.354
CB1	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.498	0.233
CB2	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.420	0.237
SY1	Li Su, Yi-Hsuan Yang	0.588	0.390
SY2	Li Su, Yi-Hsuan Yang	0.584	0.375
SY3	Li Su, Yi-Hsuan Yang	0.571	0.369
SY4	Li Su, Yi-Hsuan Yang	0.567	0.359

Multi-F0 Note Tracking - Mixed Dataset

SubID	Participants	MIREX Dataset		Su Dataset	
		Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
BW2	Emmanouil Benetos, Tillman Weyde	0.6013	0.8820	0.3190	0.7640
BW3	Emmanouil Benetos, Tillman Weyde	0.5413	0.8800	0.2855	0.7620
CB1	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.5032	0.8650	0.2267	0.7520
CB2	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.3737	0.8610	0.1572	0.8360
SY1	Li Su, Yi-Hsuan Yang	0.4786	0.8820	0.2338	0.7550
SY2	Li Su, Yi-Hsuan Yang	0.4605	0.8810	0.2278	0.8370
SY3	Li Su, Yi-Hsuan Yang	0.4616	0.8760	0.2248	0.8360
SY4	Li Su, Yi-Hsuan Yang	0.4552	0.8740	0.2223	0.8300

Multi-F0 Note Traking - Piano Only

SubID	Participants	MIREX Dataset		Su Dataset	
		Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
BW2	Emmanouil Benetos, Tillman Weyde	0.6406	0.8380	0.5000	0.8470
BW3	Emmanouil Benetos, Tillman Weyde	0.6924	0.8100	0.4751	0.8440
CB1	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.6667	0.8130	0.3582	0.8620
CB2	Chris Cannam, Emmanouil Benetos, Matthias Mauch, Matthew E. P. Davies, Simon Dixon, Christian Landone, Katy Noland, and Dan Stowell	0.4941	0.7960	0.2305	0.8220
SY1	Li Su, Yi-Hsuan Yang	0.4802	0.8090	0.3026	0.8370
SY2	Li Su, Yi-Hsuan Yang	0.4293	0.8140	0.2840	0.8310
SY3	Li Su, Yi-Hsuan Yang	0.5327	0.8150	0.2823	0.8430
SY4	Li Su, Yi-Hsuan Yang	0.4875	0.8140	0.2730	0.8350

(a) MIREX 2015. For the Multi-F0 task, the BW1 model achieves the best accuracy score, followed by the SY models (the MPE models). For the Note Track mixed dataset, model BW2 achieves the highest F-measure and for the piano music model BW3 performs best. The result holds also for the Su dataset which indicates a robustness in the BW models. The average overlap score is very even for all models, almost independently of the F-measure which indicates that this measure is not very exposing.

Multi-F0 Estimation

SubID	Participants	Accuracy	
		▾ MIREX Dataset	Su Dataset
		(2014) 0.723	(2015) 0.39
MM1	Matija Marolt	**0.537**	**0.310**
CB1	Chris Cannam, Emmanouil Benetos	0.486	0.234
DT1	Daylin Troxel	0.44	0.016
CB2	Chris Cannam, Emmanouil Benetos	0.42	0.221

Multi-F0 Note Tracking - Mixed Dataset

SubID	Participants	MIREX Dataset		Su Dataset	
		▾ Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
		(2014) 0.821	(2012) 0.893	(2015) 0.319	(2015) 0.837
DT1	Daylin Troxel	**0.712**	0.852	0.002	0.089
MM1	Matija Marolt	0.618	**0.877**	**0.315**	0.719
CB1	Chris Cannam, Emmanouil Benetos	0.503	0.865	0.228	0.731
CB2	Chris Cannam, Emmanouil Benetos	0.373	0.862	0.165	**0.803**

Multi-F0 Note Traking - Piano Only

SubID	Participants	MIREX Dataset		Su Dataset	
		▾ Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
		(2016) 0.82	(2009) 0.844	(2015) 0.500	(2015) 0.862
DT1	Daylin Troxel	**0.82**	0.796	0.004	0.000
MM1	Matija Marolt	0.754	**0.813**	**0.470**	0.819
CB1	Chris Cannam, Emmanouil Benetos	0.667	**0.813**	0.369	**0.838**
CB2	Chris Cannam, Emmanouil Benetos	0.497	0.797	0.252	0.774
KB1 (with bugs)	Rainer Kelz, Sebastian Böck	0.485	0.118	0.063	0.000

(b) MIREX 2016. For the Multi-F0 task, the MM1 model achieves the best accuracy score, followed by CB1. It can be noted that the CB1 model achieves not exactly the same results as for MIREX 2015 (Figure 13a) and MIREX 2017 (Figure 14) which it should, since it is the same model and the same dataset. This is regarded as an issue but will not be further analyzed. For Note Tracking, model DT1 stands out with the best F-measure for both the mixed dataset and the piano dataset. The scores of the Su dataset appears quite unrealistic for the DT1 and KB1 model. Some kind of bug in the evaluation can be expected here.

Figure 13:

5.6 Other models

Extending the evaluation in the thesis, in order to cover more parts of the music transcription area, a couple of models which haven't participated in MIREX were considered as well. Those

Multi-F0 Estimation

Participants	SubID	Accuracy	
		MIREX Dataset	Su Dataset
		(2014) 0.723	(2017) 0.51
Chris Cannam, Emmanouil Benetos	CB1	0.498	0.234
	CB2	0.42	0.221
Karin Dressler	KD1	0.669	0.381
	KD2	0.669	0.381
Gaku Hatanaka, Shinjiro Mita, Alexis Meneses, Daiki Miura, Nattapong Thammasan	MHMTM1	0.655	0.352
	MHMTM2	0.19	0.057
Leonid Pogorelyuk, Clarence Rowley	PR1	0.418	0.3
Katarzyna Rokicka, Adam Pluta, Rafal Rokicki, Marcin Gawrysz	PRGR1	0.408	0.062
	PRGR2	0.476	0.096
John Thickstun, Sham Kakade, Zaid Harchaoui	THK1	**0.72**	0.51
Li Su, Derek Wu, Berlin Chen	WCS1	0.593	0.357
Fuliang Yin, Weiwei Zhang, Zhe Chen	ZCY2	0.506	0.262

Task Captains: Derek Wu, Yun Hao (IMIRSEL)

Multi-F0 Note Tracking - Mixed Dataset

Participants	SubID	MIREX Dataset		Su Dataset	
		Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
		(2014) 0.821	(2017) 0.903	(2017) 0.3334	(2017) 0.841
Chris Cannam, Emmanouil Benetos	CB1	0.5029	0.865	0.228	0.731
	CB2	0.3742	0.862	0.1653	0.803
Carl Thomé	CT1	**0.7675**	0.863	0.3017	0.785
Karin Dressler	KD2	0.6969	0.886	0.3334	0.841
Leonid Pogorelyuk, Clarence Rowley	PR1	0.4497	0.863	0.2558	0.734
Katarzyna Rokicka, Adam Pluta, Rafal Rokicki, Marcin Gawrysz	PRGR1	0.4982	0.882	0.0543	0.491
	PRGR2	0.4908	0.903	0.0502	0.696
Samuel Li	SL1	0.3593	-0.039	0.2763	-0.003
Fuliang Yin, Weiwei Zhang, Zhe Chen	ZCY2	0.2256	0.78	0.2054	0.872

Multi-F0 Note Traking - Piano Only

Participants	SubID	MIREX Dataset		Su Dataset	
		Avg. F-measure Onset Only	Avg. Overlap	Avg. F-measure Onset Only	Avg. Overlap
		(2016) 0.82	(2017) 0.86	(2017) 0.5433	(2017) 0.883
Chris Cannam, Emmanouil Benetos	CB1	0.6681	0.813	0.3686	0.838
	CB2	0.497	0.797	0.2522	0.774
Carl Thomé	CT1	**0.8139**	0.786	0.4122	0.801
Karin Dressler	KD2	0.7196	0.844	0.4508	0.829
Leonid Pogorelyuk, Clarence Rowley	PR1	0.5568	0.818	0.3778	0.818
Katarzyna Rokicka, Adam Pluta, Rafal Rokicki, Marcin Gawrysz	PRGR1	0.4242	0.844	0.0261	0.51
	PRGR2	0.5529	0.86	0.04	0.883
Samuel Li	SL1	0.3122	-0.056	0.5433	0
Fuliang Yin, Weiwei Zhang, Zhe Chen	ZCY2	0.1482	0.44	0.2073	0.875

Task Captains: Derek Wu, Yun Hao (IMIRSEL)

Figure 14: MIREX 2017. For the Multi-F0 task model THK1 achieves the highest accuracy followed by the KD models. For the Note Tracking task CT1 achieves the highest F-measure for both the mixed dataset and the piano dataset followed by KD2. Just as in Figure 13b a couple of abnormal scores appears for the Su dataset for the PRGR models and the SL1. This indicates in a way that it is less trustworthy comparison than for the MIREX Dataset.

models will be briefly summarized in this section.

5.6.1 DOPT

The DOPT model was recommended in an e-mail conversation with one of the MIREX participants. The paper [46] is entitled Onsets and Frames: Dual-Objective Piano Transcription which was shortened to DOPT.

The model as described in [46], is neural network built on convolutional and recurrent layers. The input to the model consists of a spectrogram with logarithmic amplitude and with frequency spacing from the mel-scale. As defined in (4), it is a type of a logarithmic scaling which is supposed to be more adapted to the human logarithmic hearing. Each spectrum corresponds to about 13 ms audio data and consists of 229 frequency bins.

The model consists of two networks, where one network is predicting onsets and one network is predicting pitches. The onset detection network consists of a convolutional layer followed by

a LSTM layer and finally a fully connected layer which outputs a vector of 88 neurons [46]. Each neuron in the output layer corresponds to a probability that an onset has occurred for that specific tone. The architecture for the pitch estimation network is similar as for the onset detection. A convolutional layer is followed by a fully connected layer, which together with the onset prediction output is fed into a LSTM layer. The final step is a fully connected layer which outputs a vector with 88 neurons, indicating the probability of an active pitch.

In addition, a velocity estimator is added. The velocity is used to capture the speed which a piano key was depressed which is related to the loudness and articulation. As explained in [46], this estimator is built with similar structure as the onset detector and pitch estimator, but doesn't affect the note estimation more than making the resulting transcription sound more natural.

The DOPT model is created by a large group consisting of Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore and Douglas Eck at Google Brain Team, Mountain View, USA. The model is trained on the MAPS dataset [22] and is therefore supposed to only be applicable to piano data [46]. The code is written in Tensorflow.

5.6.2 PEBSI-Lite

PEBSI-Lite denotes Pitch Estimation using Block Sparsity, - Improved and Lighter.[42]. The model has not been participating in MIREX.

The model as described in [47] differs from all the other models in the sense that it is data-independent, i.e., that it doesn't use any training data at all, except for 2 regularization parameters that need to be manually set. Instead, it is built on the strong prior assumption that each frame of data consists of a stationary sequence of harmonically related sinusoids, i.e.,

$$x(t) = \sum_{k=1}^{K} \sum_{l=1}^{L_k} a_{k,l} e^{i\omega_k l t}$$

where $t = 1, ..., N$ is the time point, ω_k is the frequency of the l'th harmonic in the k'th pitch and $a_{k,l}$ is a complex-valued parameter denoting magnitude and phase. The fundamental frequencies are estimated by estimating the $a_{k,l}$-parameters for a grid of candidate fundamental frequencies. This is done in [47] by minimizing the convex optimization problem

$$C(\Psi) = \frac{1}{2} \sum_{t=1}^{N} \left| y(t) - \sum_{p=1}^{P} \sum_{l=1}^{L_{max}} a_{p,l} e^{i\omega_k l t} \right|^2 + \lambda_1 \sum_{p=1}^{P} \sum_{l=1}^{L_{max}} |a_{p,l}| + \lambda_2 \sum_{q=1}^{PL_{max}} |a_{q+1} e^{-\varphi_{q+1}} - a_q e^{-\varphi_q}| \quad (45)$$

where Ψ represent all parameters in the model, except for λ_1 and λ_2 which are regularization parameters that are set manually or using cross-validation. The cost function consists of three parts. The first is an error score which becomes as small as possible if the signal is perfectly matched. The second part is a penalty on the amplitude parameters which push down amplitudes which only contribute marginally. The third part is built on the idea that the spectral envelope is smooth. Thus, two amplitudes of adjacent overtones are penalized if the absolute difference is large.

[42] Hence the model has nothing to do with any type of soft drink.

The model considers 10 ms audio data at each time. For each time sequence an ADMM optimizer is used on (45). This is made independently of the adjacent time sequences [47]. The model is typically not rounding of the fundamental frequency estimate to the closest tone, which most part of the other models does.

The model is created by Ted Kronvall, Filip Elvander, Stefan Ingi Adalbjornsson and Andreas Jakobsson at Lund University. The regularization parameters were trained on Monte Carlo simulation of signals with different fundamental frequencies and different signal-to-noise ratios. The code was written in Matlab and it can be mentioned that it is very time consuming, due to the need to solve an advanced optimization problem for every 10 ms data.

5.6.3 ESACF

ESACF denotes the Enhanced Summary Autocorrelation Function, and has not been participating in MIREX.

ESACF is the oldest model considered in this thesis, as it was published in the year 2000 in [48]. The model divides the signal into two channels, one being high pass filter for frequencies above 1000 Hz, and a low pass filter for frequencies below 1000 Hz. The Fourier transform is then performed on the two filtered signals. The absolute value of the transforms is summed together and the inverse Fourier transform is applied [48]. This yields the summary autocorrelation function (SACF). The next step in the model is to determine which of the resulting peaks correspond to fundamental frequencies. The number of active fundamental frequencies is manually fixed. The model finds the strongest peak which is most probable to correspond to a fundamental frequency. All frequencies which appear at multiples (up to a set parameter) of this frequency are extracted and then the highest of the remaining peaks are considered. This is repeated until the set number of fundamental frequencies are found [48].

The ESACF model is written by Tero Tolonen and Matti Karjalainen at Helsinki University of Technology, Finland. The recommended parameter settings were found by testing on real data[43]. The code was written in Matlab and is very fast. In order to evaluate ESACF on the test dataset, the number of instrument sources were manually set individually for each test file. For the piano music files, that manually set number of tones was the maximum of the simultaneously played tones.

5.6.4 Deep Complex model

This model differs in many ways from the above mentioned models. The model is presented in a paper [20] on complex-valued neural networks where the main focus is on the mathematical framework of implementing operations on complex-valued numbers in neural networks. There are a couple of applications of complex-valued neural networks presented where automatic music transcription is one of them. It can therefore be assumed that the authors haven't put all their effort in tweaking and optimizing the model.

In the paper, four different models are presented where the model called Deep Complex performed the best. Deep Complex is influenced by a deep neural network model originally created

[43]The data which was referred to in [48] could not be found, probably since the article has become a few years old.

for image recognition [49]. In the paper the model was compared to a real-valued deep neural network model with similar architecture. The result was that the complex-valued model outperformed the real-valued one. Until recently, complex-valued neural networks have been an unmentioned topic in the machine learning community. But recent research has shown that complex-valued neural networks outperform the earlier real-valued state-of-the-art neural networks in many fields, where one example is hand written digit classification [1]. This invites to investigate the potential of complex-valued neural networks in polyphonic music transcription as well and compare it to the acknowledged models.

It is not unrealistic that a complex-valued neural network would outperform a real-valued counterpart. Transforming the music signal to any amplitude-frequency representation, by for example the Fourier transform, the transformation initially appears on comple-valued form consisting of both amplitude and phase. A complex-valued signal seems complicated for a human to interpret, and perhaps for this reason, the magnitude spectrum is considered in almost all music transcription models[44]. But there is an information loss by applying the absolute value. By keeping the complex-valued representation, information might be gained. This gained information, which is much easier for a computer to deal with than for a human, could possibly improve the estimates.

The Deep Complex model in [20] estimates active pitches at independent time segments. The input is the DFT from the FFT of a time segment of about 0.37 seconds, consisting of complex-valued data with 0.012 seconds overlap. According to the authors [20], the core of the Deep Complex model is its 6 convolutional networks in 1 dimension which convolve $3-6$ adjacent complex values, as detailed in (34). A convolution is followed by a complex-valued batch normalization, a $\mathbb{C}ReLU$ (35), as activation function and an average pooling. The dimensionality is decreasing because of the convolution and the pooling but for each convolution the number of filters is doubled, implying a high number of parameters[45]. The model ends with two fully connected layers where the last one uses a sigmoid function to get an output layer of size 84 (one neuron for each possible note) with corresponding probabilities[46] [20].

The model is trained using an optimizer called ADAM on the MusicNet dataset [23], which is resampled from 44100 Hz to 11000 Hz. The network is trained by dividing the data into batches consisting of 0.37 seconds segments from each of the 321 music files in the training set. In order to cover the whole dataset, about 1000 batches are needed. A training section of 1000 batches are called an epoch, and totally the model was trained on 200 epochs. The total training time was about 12 hours[47].

The model is claimed to be state-of-the-art [20]. This turned out to be incorrect since the only comparison that is made in the paper is with the real counterpart and an example model available in [23]. In [50] the Deep Complex model is outperformed by a couple of models. The result of the Deep Complex model is only presented using the average precision metric which corresponds to the area under the precision-recall curve, see Figure 11. To make it comparable with the other

[44] All except PEBSI-Lite.

[45] A similar structure will be shown for the extension of the Deep Complex model where the model architecture is shown in Figure 15.

[46] Note that a piano contains 88 notes but only 84 tones appeared in the training set. Thus the network would not be able to train the weights for the last 4 output neurons and which thereby could be excluded.

[47] On a PC with CUDA-programmable NVIDIA GPUs.

models, a binary threshold was trained, as described in Section 3.2.5.

5.7 Proposed models - Deep Complex model with LSTM and CQT/FFT

By analyzing the Deep Complex model, Section 5.6.4, a couple of remarkable features were found which weresn't clearly motivated from a music transcription perspective. The fact that the model was written by researchers who are more focused on machine learning than music automation might explain this. In this thesis, a novel method is proposed, in which these features are changed to be more proper from a signal processing and music theoretical perspective. The features were implemented directly in the source code of the Deep Complex model, but the main part of the code was left unchanged. The shortcomings of the Deep Complex model and the changes will be described further. The parts which are not mentioned are also not changed.

First of all one may note that the resampling of the dataset is not done by a multiple of 2 since $\frac{44100}{11000} \approx 4.009$. This makes it not obvious how the resampling was performed. In order to evaluate the Deep Complex model fairly, the test data needed a similar resampling as for the data it was trained with. The resampling was performed using the Python SciPy function *signal.resample* which probably doesn't differ markable from the resampling performed on the training data. Anyway, the same resampling ratio was kept for the proposed model to being able to train it on the same dataset.

A second observation is that the input is the whole DFT from the Fourier transform of 4096 complex data values. The Fourier transform is symmetric around zero for real-valued data which means that the all the input values have a doublet. This is maybe not a very complicated problem for a neural network to learn, but the number of parameters can be reduced and the calculation speed and the risk of overfitting can be reduced. To consider all 4096 Fourier values is therefore a bad choice. The Nyquist frequency at $\frac{fs}{2} = \frac{11000}{2} = 5500Hz$ is the highest possible frequency the Fourier transform is able to identify, where fs is the sampling frequency. The amount of data values which is reasonable to consider, could instead of all 4096 values be related to the Nyquist frequency, which in this case is 2048 data values. One may also note that the highest note on a piano is C8 with fundamental frequency 4186.01 Hz, which is smaller than the sampling frequency. On the other hand, no overtones of C8 can be identified, which may not be a very big issue since such high pitches are rarely occurring[48].

Considering the proposed models, the only difference between them is that one is built on the Fourier transform (described in Section 3.1) and the other on the Constant-Q transform (Section 3.1.3). The model based on the FFT has, for each 0.37s time sequence, 2048 input values corresponding to a spectrum with equally spaced frequency steps from, but not including zero, up to the 5500Hz, i.e., the Nyquist frequency. The model based on the CQT has 504 input values corresponding to 84 pitches with $12 \cdot 6 = 72$ frequency bins in each octave. The lowest pitch which is possible to detect from the CQT is A0 at 27.50 Hz and the highest pitch is $G^{\#}7$ at 3322.44 Hz. One could claim that the CQT should cover even higher frequencies since a piano goes up to the tone C8. But the fact is that no such high pitches occurred in the training data, so the model would not be able to categorize those high tones correctly anyway.

Another observation about the Deep Complex model is that it has no recurrent layers. Since the

[48]The first overtone f_1 appears at $4186.01 \cdot 2 = 8372.02$ Hz.

model is originally built for image recognition [49], where the order of the images is randomized, recurrent layers would not be of any use[49]. But for music data, there is a string dependence on previous and following data values, which almost all models in MIREX take into account. Therefore a recurrent LSTM layer may be added to the proposed models. This LSTM layer is placed just before the final two fully connected layers. Where in the network the LSTM layer should be placed optimally was not analyzed.

For the recurrent layer to make any sense, the code of the training batches needed some correction since the Deep Complex model used one time segment from each music file in the training. The correction meant that first a music file was randomized, then the start position in that music file was randomized and the following 321 time segments were considered as one batch. By making this correction, the amount of training data was not changed but the number of music files used in each epoch could be less since two music files can be randomized in the same epoch.

One final difference between the proposed models and the Deep Complex model is the number of epochs performed. Because of the longer training time caused by the LSTM layer and the Constant-Q transform, the number of epochs for the proposed models was decreased to 100. The training time for 100 epochs was about 15 hours and because of limited time, 200 epochs were not possible to perform. This would of course give a fairer comparison to the Deep Complex model since more epochs would have improved the training. However, the first 100 epochs seemed to give an acceptable estimate.

The model architecture is similar for both models, but the different inputs change the dimensionality of the layers further in the network, due to different dimensionalities of the FFT and the CQT. The complete architecture and the number of parameters is shown in Figure 15.

One aspect which would have been desirable to change is the loss the Deep Complex model is trained to maximize the standard accuracy (44) instead of the MIREX accuracy (43). The difference is that under estimating models (which estimate too few pitches) can achieve good scores using the standard accuracy measure, while the MIREX accuracy weights the estimates more reasonably. This change was however not applied to the proposed model[50].

5.8 Summary and Trends about the models

There are some trends which can be seen in the models. To illustrate those, the models are categorized in different methodical structures. This is presented in Table 3 where the methods for both signal processing, pitch estimation and potential feature detection are presented for the different models.

For pitch estimation, the models have been categorized into probabilistic, parametric and neural network methods in Table 3. The probabilistic methods are those which are using probabilistic latent component analysis. The parametric methods are using pitch estimation functions based on parameters which either are set by training or manually. The parametric methods can also be rule-based algorithms, like iterative methods, which are dependent of parameters. Neural Network methods are methods using some kind of machine learning (such as CNN or complex-valued neural networks). The feature detection column in Table 6 indicates whether there are

[49] They are randomized such that there is no relation to the previous or following images.
[50] Also because of limited time.

Deep Complex LSTMFFT

Layer (type)	Output Shape	Param #
complex_conv_1 (ComplexConv)	(None, 2048, 32)	128
complex_batch_normalization	(None, 2048, 32)	160
activation_1 (ReLU)	(None, 2048, 32)	0
average_pooling1d_1 (Average)	(None, 1024, 32)	0
complex_conv_2 (ComplexConv)	(None, 512, 64)	3136
complex_batch_normalization	(None, 512, 64)	320
activation_2 (ReLU)	(None, 512, 64)	0
average_pooling1d_2 (Average)	(None, 256, 64)	0
complex_conv_3 (ComplexConv)	(None, 256, 128)	12416
complex_batch_normalization	(None, 256, 128)	640
activation_3 (ReLU)	(None, 256, 128)	0
average_pooling1d_3 (Average)	(None, 128, 128)	0
complex_conv_4 (ComplexConv)	(None, 128, 128)	24704
complex_batch_normalization	(None, 128, 128)	640
activation_4 (ReLU)	(None, 128, 128)	0
average_pooling1d_4 (Average)	(None, 64, 128)	0
complex_conv1d_1 (ComplexConv)	(None, 64, 256)	49408
complex_conv_5 (ComplexConv)	(None, 64, 256)	98560
complex_batch_normalization	(None, 64, 256)	1280
activation_5 (ReLU)	(None, 64, 256)	0
average_pooling1d_5 (Average)	(None, 32, 256)	0
lstm_1 (LSTM)	(None, 512)	1574912
dense_1 (Dense)	(None, 2048)	1050624
dense_2 (Dense)	(None, 84)	172116

Total params: 2,989,044
Trainable params: 2,987,524
Non-trainable params: 1,520

Deep Complex LSTMCQT

Layer (type)	Output Shape	Param #
complex_conv_1 (ComplexConv)	(None, 504, 32)	128
complex_batch_normalization	(None, 504, 32)	160
activation_1 (ReLU)	(None, 504, 32)	0
average_pooling1d_1 (Average)	(None, 252, 32)	0
complex_conv_2 (ComplexConv)	(None, 126, 64)	3136
complex_batch_normalization	(None, 126, 64)	320
activation_2 (ReLU)	(None, 126, 64)	0
average_pooling1d_2 (Average)	(None, 63, 64)	0
complex_conv_3 (ComplexConv)	(None, 63, 128)	12416
complex_batch_normalization	(None, 63, 128)	640
activation_3 (ReLU)	(None, 63, 128)	0
average_pooling1d_3 (Average)	(None, 31, 128)	0
complex_conv_4 (ComplexConv)	(None, 31, 128)	24704
complex_batch_normalization	(None, 31, 128)	640
activation_4 (ReLU)	(None, 31, 128)	0
average_pooling1d_4 (Average)	(None, 15, 128)	0
complex_conv1d_1 (ComplexConv)	(None, 15, 256)	49408
complex_conv_5 (ComplexConv)	(None, 15, 256)	98560
complex_batch_normalization	(None, 15, 256)	1280
activation_5 (ReLU)	(None, 15, 256)	0
average_pooling1d_5 (Average)	(None, 7, 256)	0
lstm_1 (LSTM)	(None, 512)	1574912
dense_1 (Dense)	(None, 2048)	1050624
dense_2 (Dense)	(None, 84)	172116

Total params: 2,989,044
Trainable params: 2,987,524
Non-trainable params: 1,520

Figure 15: The architecture of the two herein proposed Deep Complex LSTM models. The left one is the FFT model using 2048 input values and the right one is the CQT model using 504 input values. There is a repeating structure with a complex-valued convolutional layer, a complex-valued batch normalization, a ReLU activation function, and an average pooling. This structure is repeated 4 times. Then follows a similar structure but with two complex-valued convolutions. In the end there are a LSTM layer followed by two fully connected layers mapping to the 84 pitches. Except the input sizes, the only difference in the network architecture is the output shapes. For example, the output shape of $average\_pooling1d\_5$ (fourth from bottom) has shape (None, 32, 256) for the FFT model and (None, 7, 256) for the CQT. This comes from the input size and since each pooling step is halving the dimension of each feature map and each convolution step doubles the number of feature maps. The number of parameters is almost 3 million which is a lot. As can be seen a big part of those are because of the LSTM layer. Thus a Deep Complex LSTM model is more challenging to train than without the LSTM layer.

any other features, except the fundamental frequencies, which are estimated by the model.

From Table 3 one may note that all neural network models are from 2016 or 2017, except the MM model which is from 2004. That can be regarded as a trend which is a natural process due to the recent growth of machine learning. It is reasonable that the probabilistic models have feature detections, but one may also note that four of the neural network models also detects features, mainly by an onset detector. Three models are using the FFT straight off. Comparing these with the other models may indicate whether some more advanced processing of the signal should be applied.

In Table 4 the training data of all the participating models are listed together with an explana-

Model	Signal Processing	Pitch Estimation	Feature Detection	Year
ESACF	FFT	Parametric		2000
MM	Gammatone filters	Neural Network	Partial tracker, onset, repeating notes	2004
Silvet	CQT	Probabilistic	Onset, offset, beat, instrument, chord	2012
BW	ERB	Probabilistic	Instrument source, sound state	2015
MPE	Spectrum, cepstrum	Parametric		2015
PEBSI-Lite		Parametric		2015
DT	CQT	Neural Network	Onset	2016
CT	sliCQ	Neural Network	Onset	2017
DOPT	Mel scaled FFT	Neural Network	Onset	2017
KD	Multi res. FFT	Probabilistic	Tone envelope, onset	2017
MHMTM	CQT	Neural Network		2017
PR	DMD	Parametric		2017
THK	Filterbank	Neural Network		2017
DeepComp	FFT	Neural Network		2017
DeepLSTMCQT	CQT	Neural Network		2018
DeepLSTMFFT	FFT	Neural Network		2018

Table 3: Table of the basic structures in the models. Models with multiple variants such as different training data etcetera, are merged together in this table (for example PR=PR1&PR2).

tion of how all participating models were estimated.

From Table 4 it can be noted that there is a limited amount of training datasets which are used by the models. MusicNet and MIREX development set are real value data while the RWC and MAPS datasets are MIDI files which are simulated sound. About the evaluation it should be noted that the estimates from models CT1 and DT1 were received by mail after sending the test files to the model authors. All other estimates were calculated in-house.

Model	Training Data	Evaluation Test files
BW1	RWC, orchestral	Matlab
BW2	RWC, orchestral	Matlab
BW3	MAPS, piano	Matlab
CT1	Unknown	Achieved through mail
DeepComp	MusicNet, orchestral and piano	Python, tensorflow
DeepLSTMCQT	MusicNet, orchestral and piano	Python, tensorflow
DeepLSTMFFT	MusicNet, orchestral and piano	Python, tensorflow
DOPT	MAPS, piano	Python, tensorflow
DT1	MIDI files, orchestral and piano	Achieved through mail
ESACF	Unknown	Matlab
KD1	MIREX development set, orchestral	Executable file
MHMTM1	MIDI files, orchestral and piano	Python, tensorflow
MM1	MIDI files, piano	Executable file
MPE1	Unknown	Matlab
MPE2	Unknown	Matlab
MPE3	Unknown	Matlab
MPE4	Unknown	Matlab
PEBSI-Lite	None	Matlab
PR1	Unknown	Python
PR2	Unknown	Python
SilvetMF0	RWC, MAPS, Disklavier, orchestral and piano	Executable file
SilvetMF0piano	MAPS, Disklavier, piano	Executable file
SilvetNT	RWC, MAPS, Disklavier, orchestral and piano	Executable file
THK1	MusicNet, orchestral and piano	Python, tensorflow

Table 4: The training data of each model together with a description of what program the test files were estimated with.

6 Tests

One of the most interesting questions this thesis is trying to answer is: Which model is state-of-the-art in music transcription right now? The reason why this answer is not readily found on the Internet or even at MIREX, is because all music transcribing models are not evaluated at the same time or at the same dataset[51]. This thesis gathers many of the last years recent models and compares them on the same dataset. One advantage with the tests in this thesis is that the datasets afterwards will be available for the public and it will thus be possible to see exactly how each model is performing for each tone in each song[52].

In this section, the steps taken by going from estimates to a measure of the performance will be explained. First by presenting the test dataset in 6.1, then explaining how an estimate of Note Track form is converted to Multi-F0 form (and vice versa) in 6.2. Then, in Section 6.3, the methods are aligned to give outputs of the same format, and finally in Section 6.4 it is explained how the MIREX measures are calculated practically.

6.1 Dataset

There was a couple of things to consider while designing the test-dataset. The main idea was to make it representative of the MIREX development dataset. The only comparable dataset is the MIREX development dataset which consists of real instrument recordings in mono, sampled at $44,1$ kHz[53]. The MIREX development dataset consists of the fifth variation from L. van Bethoven Variations from String Quartet Op.18 N.5. The song is arranged for a woodwind quintet consisting of Bassoon, Clarinet, Flute, French Horn and Oboe. Each of the instruments are performed and recorded by itself while the performer was listening to the other parts through headphones. Finally they are mixed together. The sheet music annotations are manually transcribed and are used as the ground truth for the model [18].

For the scope of this thesis, it was not possible to record a test dataset as realistically as the MIREX development dataset. A couple of real recordings with manually annotated sheet music are available on the Internet, such as the MusicNet dataset and the Bach-10 dataset, summarized in Table 2. But since these datasets are few, there is always a risk that one of the models has used this dataset as its training set, which wouldn't give fair evaluation. Instead, inspired of the partly realistic MIDI-version of the real recording of the MIREX development dataset, MIDI files from MuseScore [51] were used as the test set. Sheet music from famous composers arranged by private persons are being shared on the MuseScore homepage and can be downloaded for free. Only sheet music which was available for commercial use has been part of the test dataset. By using MuseScore, the sheet music could be transformed to wav-files with more realistic sound than MIDI files. Also, the ground truth is completely known. Compared to sheet music annotations made manually by professionals, the MIDI file sheet music representation is actually more precise. For example, in the MusicNet dataset, there is an expected error rate of 4%. One could

[51] A model in MIREX is only compared with participating models of the same year and all other papers about music transcription which are not participating in MIREX are mainly evaluated on extracted parts of the training dataset.

[52] In Section 4.1 it was stated that the MIREX Dataset is needed to be kept secret.

[53] The MIREX development dataset is available for research. A request form was filled in and accepted by the MIREX organizers.

note that the play time of a wav-file and a MIDI file differs. By analyzing the music files, the playtime of the MIDI file ends directly when the sound ends while the wav-file has some quite seconds in the end. This indicates that the MIDI file is the true ground truth and therefore this issue is disregarded.

The test dataset consists of 11 music files from MuseScore and 1 music file from the Bach-10 dataset. The Bach-10 file was added to investigate how the models were performing on real recorded music and not only simulated music. By a quick analysis of the evaluated models, none of them seemed to have been trained on the Bach10 dataset which approves the use of it in the testset[54]. The test dataset was chosen to be as wide and general as possible but still similar to the MIREX test set. It needed to contain at least some ensemble music and some piano music since that is what the MIREX Dataset does. The test dataset is restricted to classical music and hymns with no percussion or singing. There are woodwind compositions and piano music similarly to the MIREX test set, but some other instruments are investigated here as well such as the guitar, the violin and a string quartet. The pace of the tones varies from the different music files as well as the number of tones played at the same time, so each of the music files in the test set investigates how well each model manages a certain type of music for a certain instrument. The test files were chosen to have a similar length of time and no replays[55]. The test dataset is stated in Table 5.

One thing to notice is that the wav-files from MuseScore were in stereo, while the wav-file from Bach-10 were mono. The majority of the models were able to handle multi-channel recordings, while some models (PEBSI-Lite, ESACF and THK1) could only cope with single-channel inputs. In those cases, the two channels were mixed together by summing the signals.

6.2 Translation between formats

The two different formats for Multi-F0 and Note Track, illustrated in Section 4.2 and 4.3, fill different purposes but are still similar. Each model either has an output similar to the Multi-F0 format or the Note Track format[56]. To be able to evaluate each model for both the tasks a translation between the different formats was needed to be implemented.

From Note Track to Multi-F0 is a straight-forward translation. In each row of the Note Track output there is a tone represented by an onset, an offset, and a frequency. The translation algorithm does the following: For each tone, each of the time bins between the onset and offset creates a pair for that frequency. Finally, all frequencies which are paired with each time bin are stacked in a row, resulting in the active frequencies for that time bin such that each time bin corresponds to none, one, or multiple frequencies.

Going from Multi-F0 to Note Track is a bit complicated. In the Multi-F0 output there are time bins with corresponding active frequencies. The translation algorithm does the following: Start-

[54]Since the Bach10 dataset only consists of 10 music files, it would for example not be reasonable to use the Bach10 to train a neural network. Therefore the risk that Bach10 would occur as training dataset for some of the models is low.

[55]A similar time was needed such that they wouldn't need a weighting when the results are added together. No replays should appear since two equal parts of a song would give a very similar estimation which would not add anything to the evaluation.

[56]except MPE which gave both output formats directly and the Deep Complex model (as well as the Deep Complex LSTMCQT and Deep Complex LSTMFFT) which gave another type of output.

Music piece name	Composer	Arranger	Short name	Instruments	Numb. of instruments	Song length	Song speed and style
Nun bitten wir den heiligen Geist (from Bach10)	J.S. Bach	-	mix1	Violin, clarinet, saxophone, bassoon	4	0:37	Hymn, long tones
"Graduale" from th Requiem in C Minor No.2	L. Cherubini	Mike Magatagan	mix2	Flute, oboe, clarinet, bassoon	4	01:13	Slow, long tones
Tochter Zion	G. F. Händel	User: Oboeli95	mix3	Flute, oboe, B^b-clarinet, french horn, bassoon	5	01:03	Medium, long & short tones
Prelude: "Helft mir, Gottes Güte preisen"	J.S. Bach	Mike Magatagan	mix4	Flute, oboe, A-clarinet, bassoon	4	01:12	Fast, short tones
Hark! The Herald Angels sing	F. Mendelssohn	User: dan.ricci.37	mix5	Violin × 2, viola, flute, B^b clarinet, bassoon, B^b trumpet × 3, C tuba	10	0:39	Hymn, slow
B flat scale	-	User: Hersey1754	piano1	Piano	1	0:33	Monophonic, long tones
Lullaby	J. Haydn	Don Moss	piano2	Piano	1	01:12	Slow, long tones, chords
Sonata No.2 4^{th} Movement Opus 35	F. F. Chopin	User: ClassicMan	piano3	Piano	1	01:25	Fast, short tones
Trepak (Russian Dance) From the Nutracker Ballet Suite Op. 71A: II. C.	P. I. Tchaikovsky	User: Alphonse Gayloa Samson	piano4	Piano	1	01:08	Fast, short tones, chords
Adantino Op.241 No.20	F. Carulli	User: Marieh	guitar1	Guitar	1	0:52	Medium, long & short tones
Divertimento in F K138 III. Presto	W. A. Mozart	User: AR-Boyes	violin1	Violin	1	0:22	Medium, long & short tones
Alla Rustica III. Allegro	A. Vivaldi	User: pribylova	strings1	Violin × 3, violincello	4	0:52	Monophonic, short tones

Table 5: Summary of the test dataset. All music pieces are used as -wav format. The top dataset (short name mix1) is a real recording from Bach10 dataset [24]. The rest are simulated sound from MuseScore files [51].

ing at the first time bin, the first frequency is considered as a part of the first tone. The onset of this tone is the first time bin. In the following time bins, this frequency is searched for. If the frequency is found, the tone is considered as still active and the next time bin is considered and so on. If the frequency is not found, the previous time bin is considered as the offset of the tone. Then one goes back to the second frequency of the first time bin and repeat the process. This is repeated until for all frequencies of the first time bin. For the following time bin one searches for frequencies which were not active in the previous time bin. For each new frequency the procedure of finding an offset is done.

The described translation of Multi-F0 to Note Track does only work for a Multi-F0 representation where frequencies has been truncated to the closest tone. For example, if a model estimates a frequency of 225 Hz the closest tone is A3 with a pitch of 220 Hz ($A^{\#}3$ is the second closest with tone with pitch 233.082 Hz). Most of the models do this truncation automatically, but some don't[57]. With the above algorithm, almost all tones appear only for one time bin for the models without this automatic rounding. Since this is not a fair way of comparison, the translation considers all frequencies which are within $\pm 3\%$ from a tone to be rounded to the corresponding pitch.

6.3 Conversion to MIREX format

There are very strict rules in the MIREX format for evaluation. In order to evaluate the estimates they need to be given on a consistent fashion[58]. For most of the MIREX models, the output of the given source code was correctly, but some models needed minor modifications. This section will describe which modifications were needed to evaluate the models. The modifications should not affect the performance of the models, but to recreate the presented results, similar changes are needed.

Some methods yielded more accurate time or frequency estimates than a two decimal precision, which is the MIREX standard for both Multi-F0 and Note Track. In those cases, the values were rounded to two decimals[59]. Another case that was handled was when some models didn't output a text file. For the outputs in excel format, the estimates were just copied into a txt-file. For the models which gave a matrix or cell-array output in Matlab, the output was converted into a text file. Two of the models gave MIDI files as output. Those cases were converted in Matlab with [52] which transforms the MIDI file to a variant of Note Track format. The transformation includes the onset and offset as well as the MIDI number, the velocity and instrument number. A MIDI number corresponds to a certain pitch and can be translated to a frequency as in [9] by

$$f_n = 2^{\frac{n}{12}} \cdot 440Hz \tag{46}$$

where n is the MIDI number and f_n is the corresponding frequency. By extracting the onset, offset, and MIDI number, and transforming the MIDI number to frequency, the Note Track format was obtained. This conversion was also used when transforming the MIDI files in the test

[57] It is common for music transcription models to automatically round to the closest tone since a piano roll representation usually is what is wanted. Anyway, there are models like PEBSI-Lite and ESACF which just estimate active frequencies independently of what frequencies can be expected from musical instruments.

[58] One format for Multi-F0 and one for Note Track as described in Section 4.2 and 4.3.

[59] In MIREX it is accepted to submit estimates with more than 2 decimals. But when various formatting files were written for this thesis, this was not considered. In order to not rewrite all the formatting files the rounding solution was the most efficient.

dataset to the ground truth Note Track and Multi-F0 format.

The most demanding conversion was for the Deep Complex models[60]. The output was a $n \times 84$ matrix with probabilities for each of the 84 possible tones in each of the n time bins[61]. To determine which tones to be considered as active, a binary threshold as in Section 3.2.5 was implemented. Probabilities which appeared higher than the threshold were considered as active tones. The threshold was trained for each of the Deep Complex models by maximizing the MIREX accuracy (43), for the Music net test set using Matlab to minimize the negative accuracy. Using the threshold the active tones were extracted for each time bin and then transformed to pitches[62]. The output finally achieved the Multi-F0 format.

Presenting the output of the Deep Complex model as probabilities is not a bad idea, since the average accuracy could be calculated from the precision recall-curve using the probabilities, Section 3.2.5. However, a presentation with a Multi-F0 or Note Track format is more intuitive and closer to a usable product. That is probably the reason why a probability based output is not considered in MIREX.

As summary of the output formats of the different models, consider Table 6. To get all models of both Multi-F0 and Note Track format, the above presented conversions and the presented translations in Section 6.2, were applied for the models which didn't output both formats directly.

6.4 MIREX Evaluation

The output as Multi-F0 or Note Track format from the models are evaluated with different measures, as described in Section 4.2 and 4.3. The measures which are considered in this thesis are, for Multi-F0, the 3 measures precision, recall and accuracy and similarly for Note Track, the 6 measures precision, recall and F-measure both with and without the offset criterion[63]. To calculate these measures, the Python library *mir_eval* version 0.4 was used from [29].

One advantage with using an existing evaluation library is that the results can easily be recreated. Another is that this library is very user friendly. The library is also transparent such that it is robust to a shift between the estimation and the ground truth[64]. The disadvantage is that the evaluation is not identical to how MIREX is evaluating the different measures. This would have been possible to achieve by proposing a new evaluation program. Some disadvantages are that bugs can be hard to identify and if any changes are made in the library by its authors, the exact same results might not be possible to recreate[65].

For evaluating the Multi-F0 the *multipitch* function [29] was used, which doesn't differ particularly from the MIREX evaluation. The number of matching frequencies for each time bin between the estimated Multi-F0 output and the ground truth on Multi-F0 form is calculated and from this, precision, recall and accuracy are calculated. If the matching is not ideal ($accuracy \neq 1$ in

[60] Deep Complex model, Deep Complex LSTMCQT and Deep Complex FFT.

[61] The size was motivated in Section 5.6.4.

[62] Recall that this is a $n \times 84$ matrix where 1 indicates an active tone and 0 indicates an inactive tone. The transformation from this matrix was done with a similar formula as for the MIDI number, (46).

[63] Accuracy is referred as the MIREX accuracy, (43).

[64] To avoid implementing this, saved the author many hours of work.

[65] Some bugs were identified in the library due to weird test results. These bugs were possible to work around by adjusting the data, but there might be smaller bugs in the library which weren't found in this thesis.

Model	Multi-F0 Format	Note Track Format	MIDI Format	Other
BW1	x			
BW2		x		
BW3		x		
CT1				Detailed version of Note Track
DeepComp				Note Probabilities
DeepLSTMCQT				Note Probabilities
DeepLSTMFFT				Note Probabilities
DOPT			x	
DT1				Note Track in Excel (3 decimals)
ESACF				Multi-F0 in Matlab Matrix
KD1				Note Track (6 decimals)
MHMTM1	x			
MM1			x	
MPE1	x	x		
MPE2	x	x		
MPE3	x	x		
MPE4	x	x		
PEBSI-Lite				Multi-F0 in Matlab cell array
PR1	x			
PR2		x		
SilvetMF0	x			
SilvetMF0piano	x			
SilvetNT		x		
THK1	x			

Table 6: "x" corresponds to the format of the model output. There are 4 models (MPE 1-4) which output was not modified at all by the author. The other models were modified by a translation or conversion (or both).

(43)), the time bins of the estimate are resampled using nearest neighbour to achieve the best possible match of the estimate and ground truth. The frequencies of the estimate, corresponding to the time bins which appear outside the ground truth time bins after the resampling, don't contribute to the evaluation.

The evaluation of the Note Track is done by the *transcription* function [29] which is not as intuitive as for Multi-F0. The evaluation method counts how many of the estimated notes which match the ground truth and how many which don't. Based on these counts, the precision, recall, and F-measure are calculated. The criteria for a tone to be considered correct is (from Section 4.3) that the onset is within ±50 ms, the frequency is within ±3% (a quartertone), and the offset needs to be within a 20% range, from the ground truths[66]. In this evaluation library, the offset criteria is slightly changed to be within 20% range from the ground truths' offset or within ±50 ms, whichever is larger[67]. Another note to make is that the evaluation method uses a bipartite graph matching to find the optimal paring[68].

The library is used both to evaluate the Note Track estimation with the offset and without the offset criterion[69]. Compared to the MIREX evaluation method, this library is expected to give a higher score of the precision, recall and accuracy by about $1\% - 2\%$[70]. This is not perfectly desirable but since the difference is not considerably large it will not make that big difference. Also, all models will be evaluated with the same conditions, thus this will not affect the result.

A couple of smaller modifications are necessary to address, in order for the functions in the library to compile. Some of the estimates got negative time bins, which could for example look like

```
−0.02
−0.01   220.00
 0.00   220.00
  ...
```

which were not accepted. In these cases, the time bins < 0 were deleted. Another error occurred for some of the Note Track estimates when a tone only appeared for a single time bin (onset and offset appeared on the same time bin). Since it is technically impossible for a tone to hold on for less than 0.01 seconds, those tones were deleted from the Note Track estimates to make the library function compile.

[66] The "within" is clarified as -50 ms $\geq f_0 \leq 50$ ms for example. I.e., all "within" measures includes the limit.

[67] There is contradictory information about the criterion for the offset on the MIREX homepage [18], that's why this confusion may have appeared.

[68] This is done since each estimated tone can only be paired with one true tone.

[69] The score without the offset criterion is always higher ($\geq$) than the score with the offset criterion

[70] According to tests by the authors of the library [29].

7 Results

The results of this thesis are in one way presented as plots of the models' estimates of test files combined with the ground truth. Another way the result is presented is by the general measures of precision, recall and accuracy/F-measure which are shown in diagrams for all models simultaneously. A third presentation of the results would be to listen to the MIDI-translations of the model outputs. This would give a quite proper measure of how well a model is performing according to how a user would perceive it. Because of natural causes this measure will be disregarded in the thesis.

In this section first the diagrams of the Multi-F0 evaluation is presented Section 7.1, followed by the same for Note Track 7.2. Afterwards some of the estimates will be plotted in Section 7.3 and finally the results are analyzed in Section 7.4.

7.1 Results of Multiple Fundamental Frequency Estimation

First the mean of the test files is presented for all files, for all ensemble files (called mix files) and all piano files. Since the results differ from each test file, the diagram for each of the 12 test data files are presented as well.

7.1.1 Averaged Multi-F0 Results

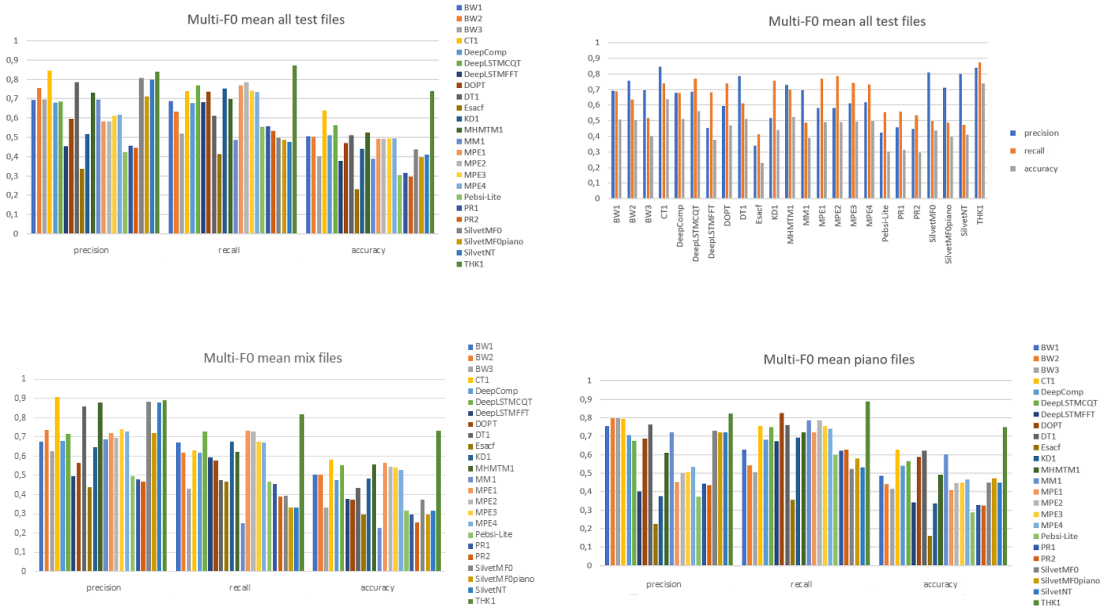

Figure 16: Compilation of all mean values for the Multi-F0 estimates

7.1.2 Single Test Files Multi-F0 Results

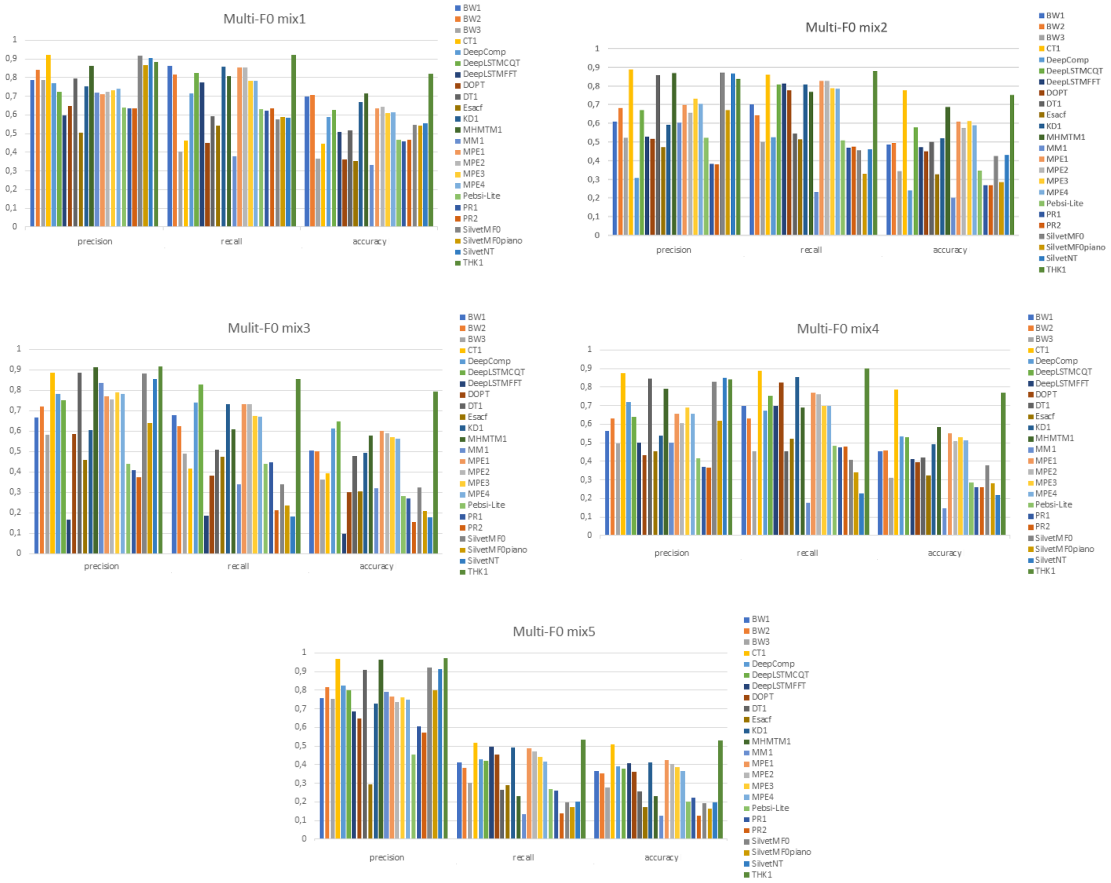

Figure 17: Compilation of all Multi-F0 estimates on the ensemble files in the test dataset

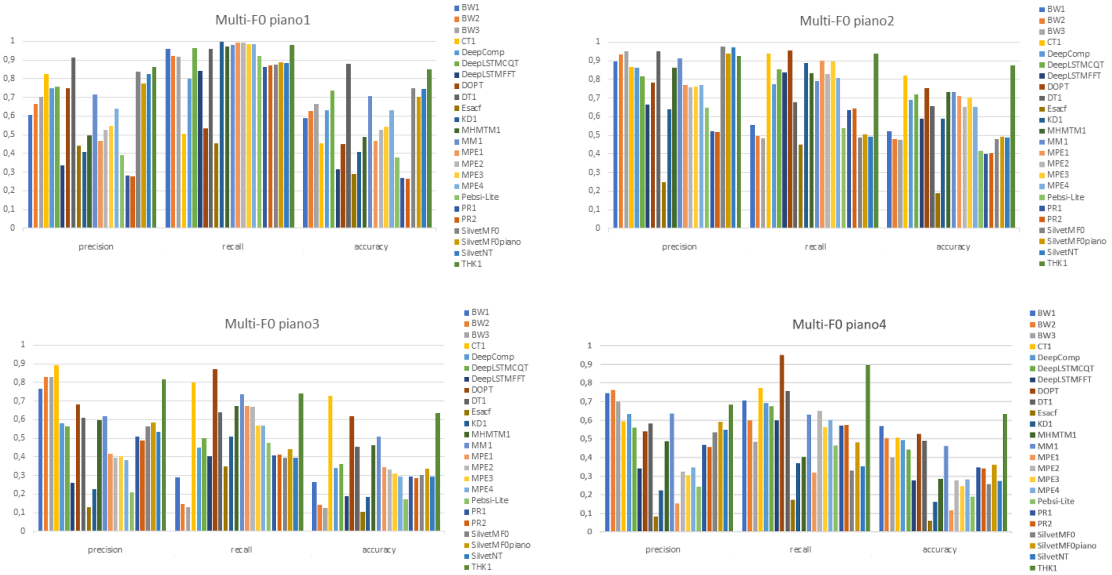

Figure 18: Compilation of all Multi-F0 estimates on the piano files in the test dataset

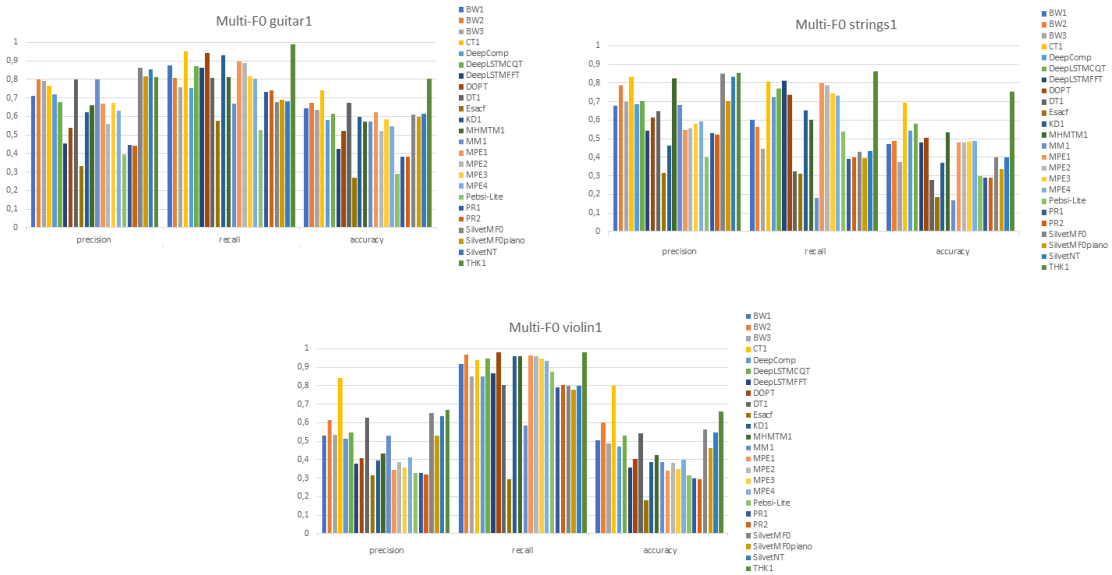

Figure 19: Compilation of all Multi-F0 estimates on the other files in the test dataset

7.2 Result of Note Track

7.2.1 Averaged Note Track Results

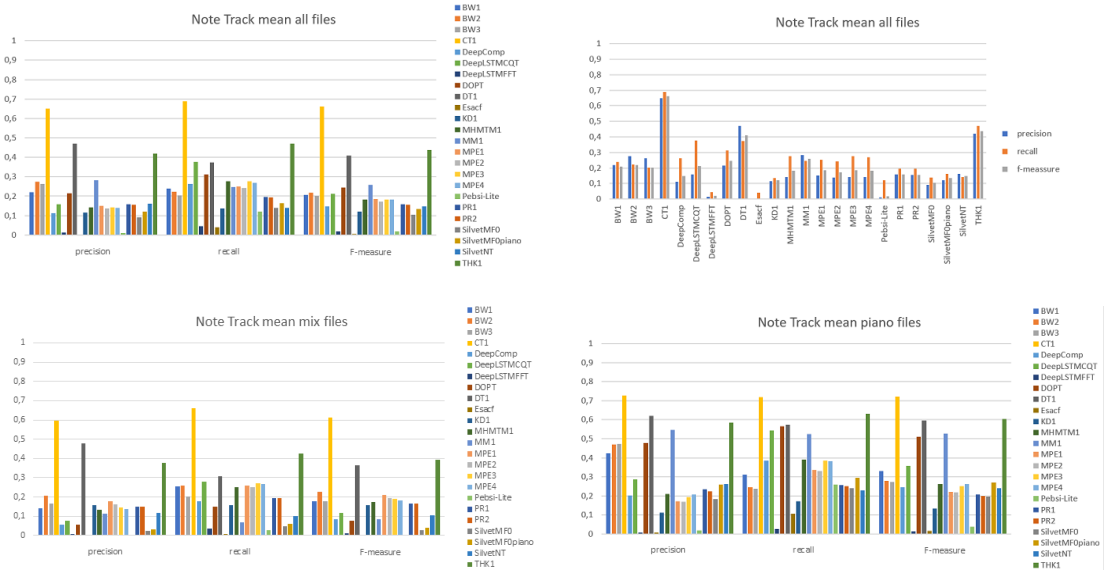

Figure 20: Compilation of all mean values for the Note Track estimates

7.2.2 Single Test Files Note Track Results

Figure 21: Compilation of all Note Track estimates on the ensemble files in the test dataset

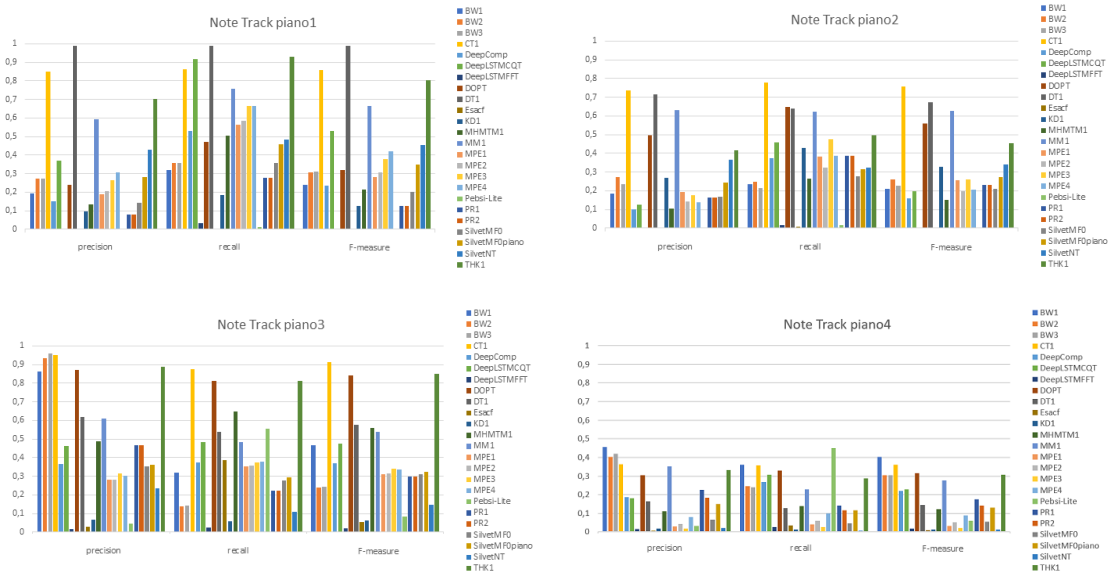

Figure 22: Compilation of all Note Track estimates on the piano files in the test dataset

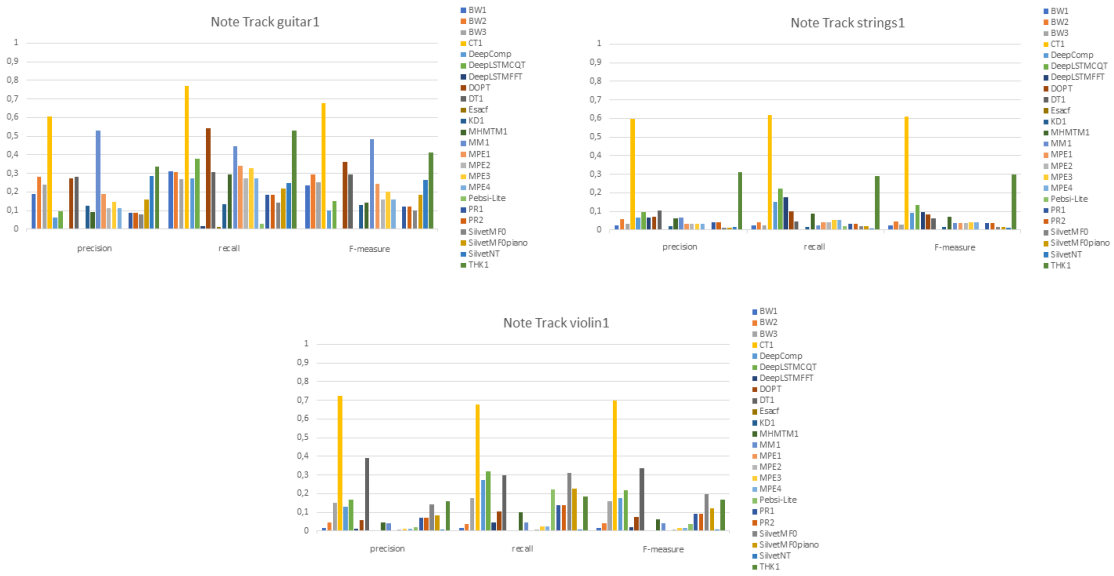

Figure 23: Compilation of all Note Track estimates on the other files in the test dataset

7.2.3 Note Track Without Offset Criterion

Figure 24: Compilation of all mean values for the Note Track without offset criterion estimates

7.3 Plots of estimates

The results presented in the figures in Section 7.1 and 7.2 are the main results of this thesis. In order to analyze the results and to find out the types of errors behind the performance metrics, a subset of results will be visualized in time-fundamental frequency plots. Visual inspection makes it possible to connect the different methodical structures in the models to their performance. Because of the large number of test files (12) and models (24), the results of all test files of all models ($12 \cdot 24 = 288$) cannot be presented. Instead, a selection of these, where specific results are of more interest, will be included.

In this section a lot of plots will be presented. Each plot will include the ground truth of the whole part of one test file (a whole song) and the estimate from a single model. The plots are of the Multi-F0 representation, where the estimated pitches for each time bin of 10 ms (100 samples per second) are visualized. For each estimation, the plot will indicate a true positive pitch (a correctly estimated pitch within a quartertone from the ground truth) by a large black dot. A false positive pitch (incorrectly estimated pitch) will be indicated by a pink cross. Since a tone in general lasts longer than 10 ms, the dots will be stacked such that they will look like a line. The ground truth is a thin colored dot where the color indicates which instrument that was playing that pitch[71].

The representation will give a fairer indication of the Multi-F0 score than the Note Track score. To include some indications of the Note Track score, each pitch which is incorrectly estimated but within the time interval of ± 50 ms of a ground truth pitch, will be indicated by a yellow square. Those incorrect frequencies may in the Note Track representation, be included in a correctly estimated tone. A final indicator is if an estimated pitch appears between a quartertone and a semitone from a ground truth pitch. Those pitches will be marked by an orange star and indicates that the signal processing has failed. Thus, the incorrect tones marked by a pink cross will mainly indicate where the pitch estimation has failed, that is a false positive. Another type of error is when a ground truth tone is not estimated, indicated by a colored line which has no black line behind, i.e., a false negative. The plots which follow are analyzed further in Section 7.4.

[71] Note that the models only estimate pitches and not which instrument playing each pitch. However, the ground truth information about the instrument can be used for analyzing the result.

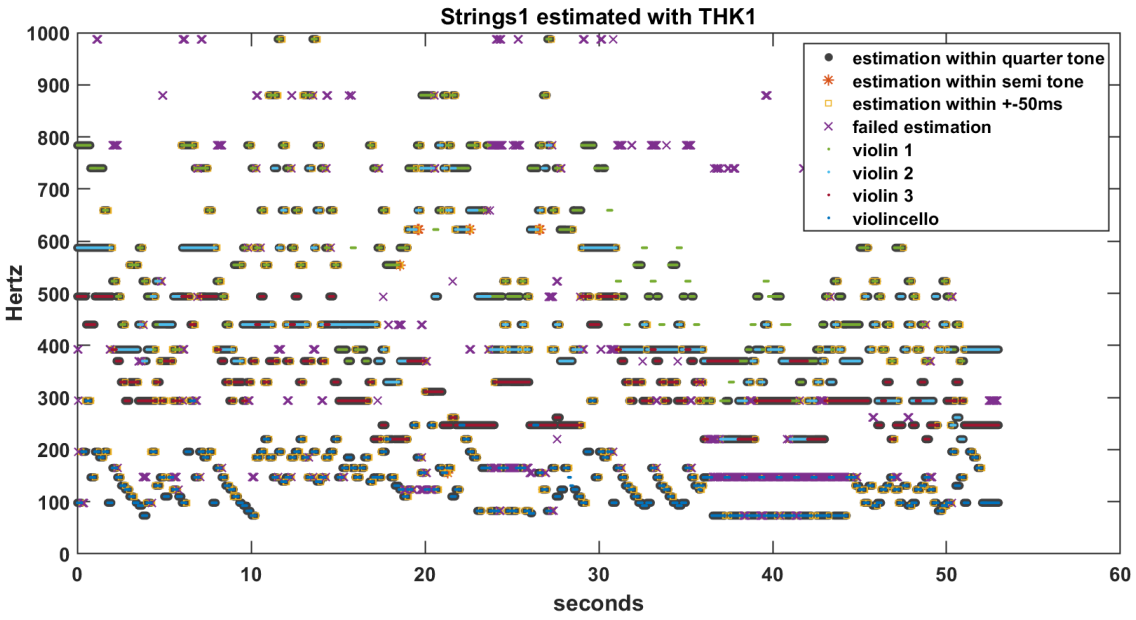

Figure 25: Alla Rustica by A. Vivaldi. Performed by a string quartet. Multi-F0 scores: Precision = 0.85294, recall = 0.86435, accuracy = 0.75224. Note Track scores: Precision= 0.31186, recall = 0.28840, F-measure = 0.29967. The string ensemble was in general quite complicated to estimate especially for the Note Track task. THK1 got the highest accuracy score for Multi-F0 and the second highest F-measure score for Note Track. The precision and recall are well balanced which is optimally in achieving a high accuracy and F-measure. One can see that the model estimates pitches of both higher and lower frequencies with a similar specificity. The only structural error found, is for the violincello between 24-27 seconds and 37-45 seconds when fast octave jumps occurs.

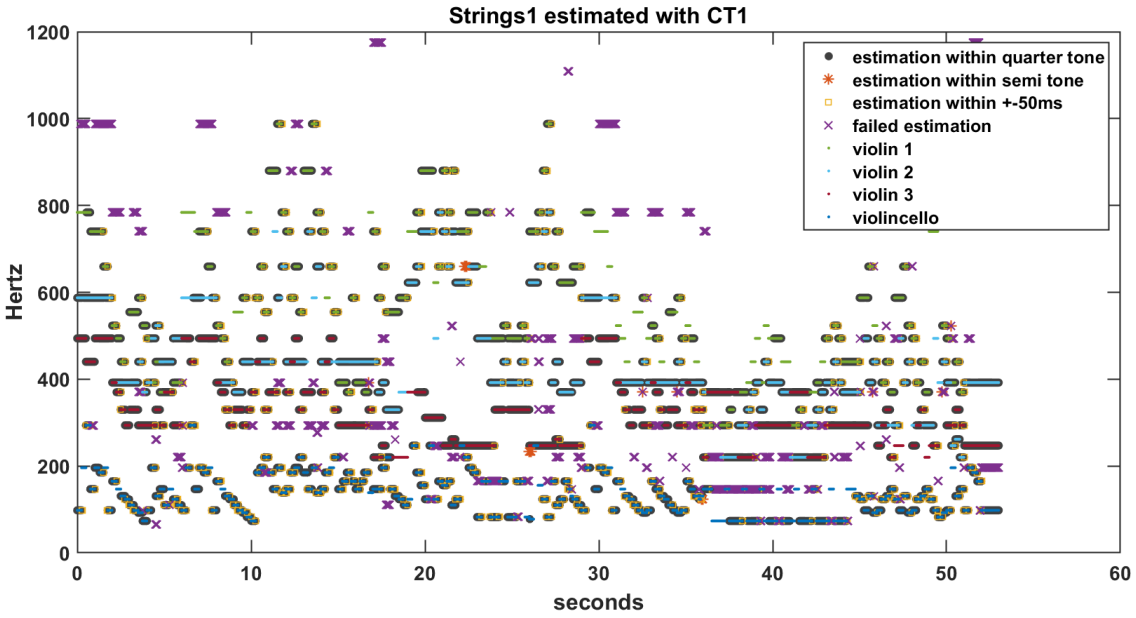

Figure 26: Alla Rustica by A. Vivaldi. Performed by a string quartet. Multi-F0 scores: Precision = 0.83192, recall= 0.80690, accuracy = 0.69380. Note Track scores: Precision = 0.59639, recall = 0.62069, F-measure = 0.60829. From the plot it looks much similar as the estimation of the THK1 model in Figure 25. A higher rate of false negatives is estimated by CT1 than THK1. Due to a higher Note Track score, CT1 is concluded to hit the onset and offset better than THK1.

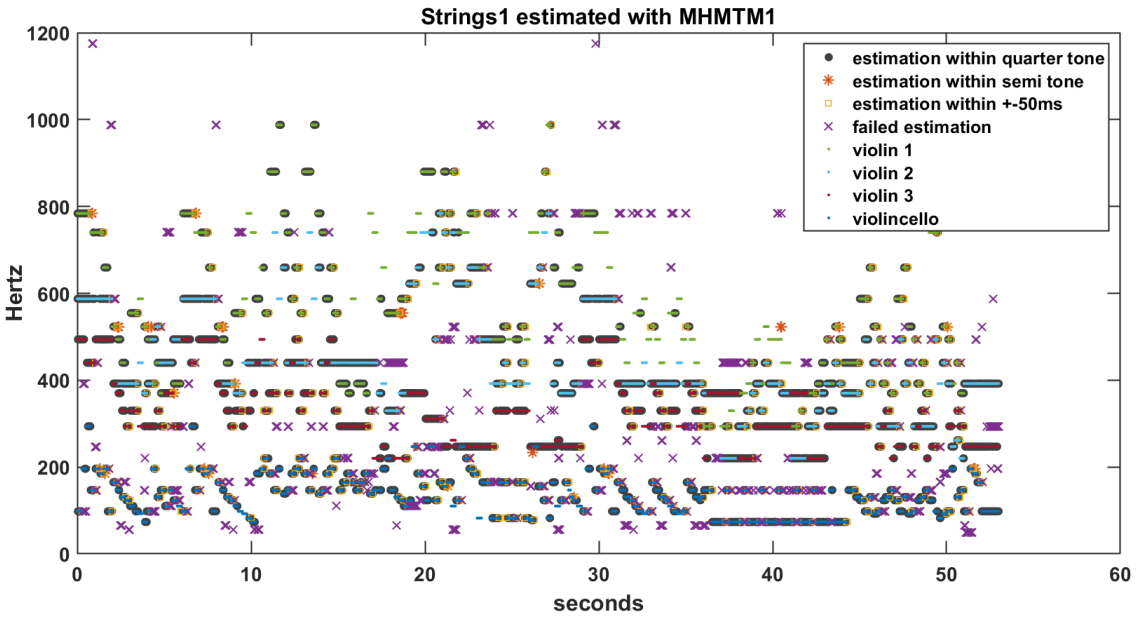

Figure 27: Alla Rustica by A. Vivaldi. Performed by a string quartet. Multi-F0 scores: Precision = 0.82711, recall = 0.60387, accuracy = 0.53619. Note Track scores: Precision = 0.06147, recall = 0.08777, F-measure = 0.07230. The MHMTM1 performs relatively well but it's a clear difference from THK1 and CT1. By studying the lower frequencies a lot of false positives (pink crosses) are found, while in higher frequencies, a lot of false negatives are estimated (thin colored lines with no thick black line behind).

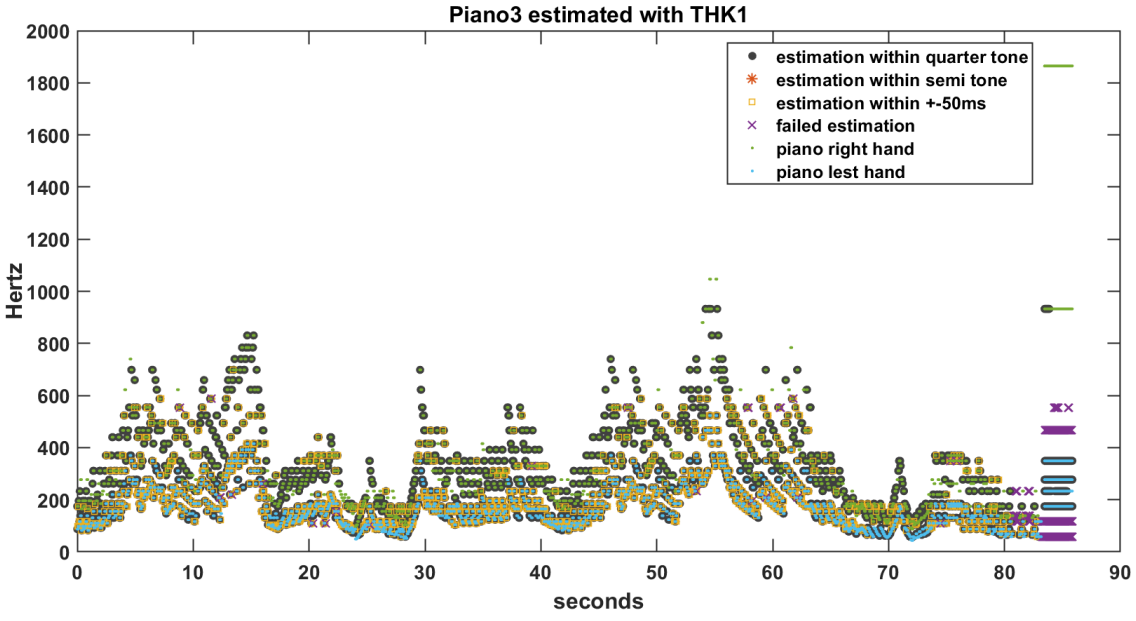

Figure 28: Sonata No.2 by F. F. Chopin. Performed on piano. Multi-F0 scores: Precision = 0.81629, recall = 0.74094, accuracy = 0.63505. Note Track scores: Precision = 0.88862, recall = 0.81195, F-measure = 0.84855. The estimation of this high speed piano melody gets very well estimated by THK1. Very few false negatives occur, only some of the highest tones from what can be seen in the plot. On the last chord at 83-86 seconds, a couple of false positives appears.

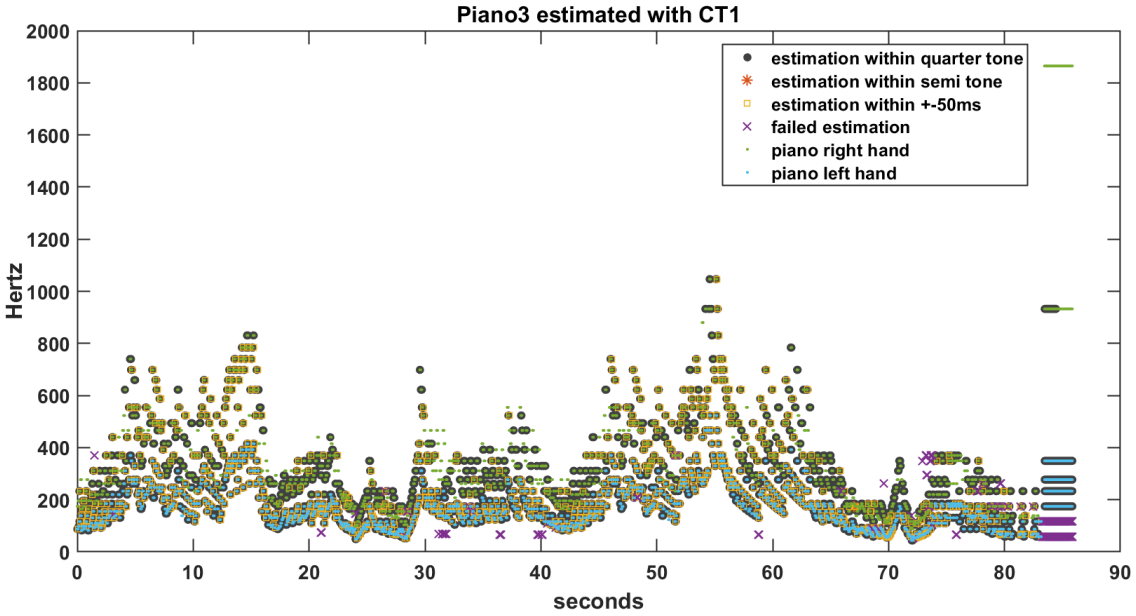

Figure 29: Sonata No.2 by F. F. Chopin. Performed on piano. Multi-F0 scores: Precision = 0.89047, recall = 0.79800, accuracy = 0.72667. Note Track scores: Precision = 0.95129, recall = 0.87500, F-measure = 0.91155. Also a well performed estimation by CT1. A lot of yellow squares, similar as Figure 28, indicating a minor time resolution error which doesn't affect the Note Track score. Also similarly, the final chord consists of a couple of errors.

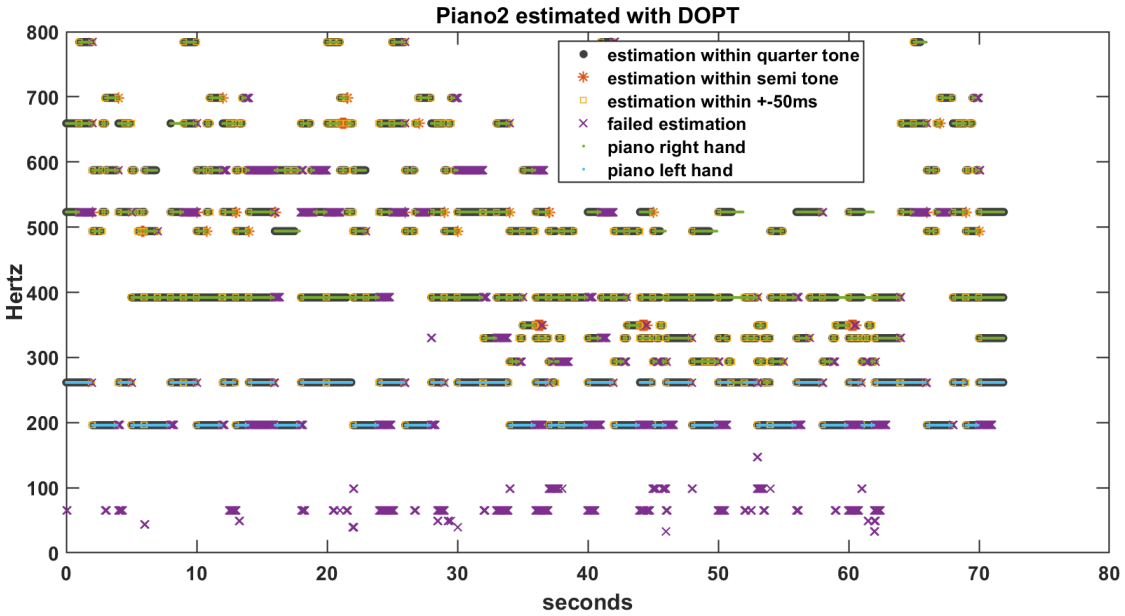

Figure 30: Lullaby by J. Haydn. Performed on piano. Multi-F0 scores: Precision = 0.78113, recall = 0.95356, accuracy = 0.75251. Note Track scores: Precision = 0.49533, recall = 0.64898, F-measure = 0.56184. A well performed estimation with a very high recall in the Multi-F0 task. A couple of tones seem to be estimated as false positives around 80 Hz, this is odd since they don't seem to appear at the half frequency of any ground truth, $\frac{1}{2}f_k$. Most probable, they are related to $\frac{1}{3}f_k$ which would be quite easy to prevent by tweaking the model. Other failed estimates seem to appear after the offset of the true tone.

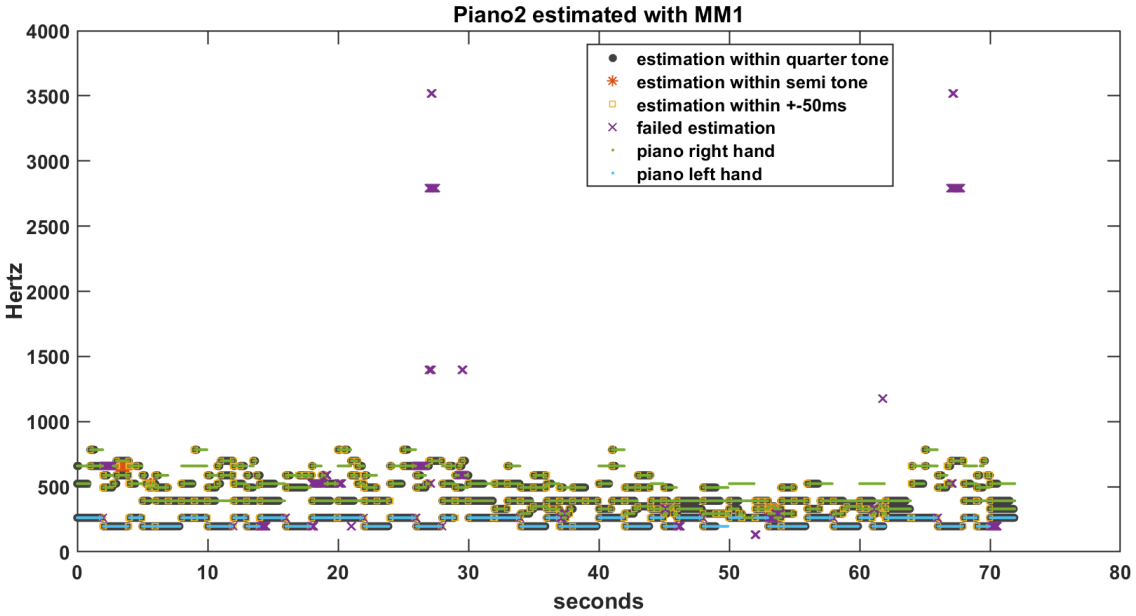

Figure 31: Lullaby by J. Haydn. Performed on piano. Multi-F0 scores: Precision = 0.91234, recall = 0.78978, accuracy = 0.73407. Note Track scores: Precision = 0.62963, recall = 0.62449, F-measure = 0.62705. This is the same test file as in Figure 30, but in another scale in order to include the failed estimates of the high frequencies. MM1 performs a bit different from the DOPT, but also with a high accuracy and F-measure. There are very few false positives, but more false negatives estimated by the model. This issue is most salient for the frequencies around 600 Hz.

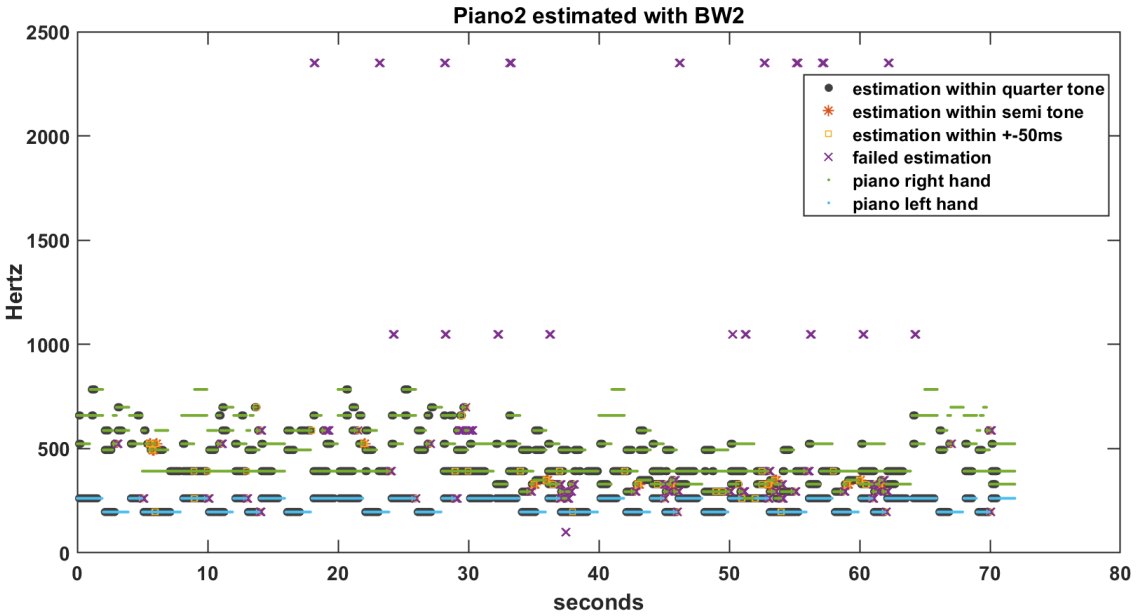

Figure 32: Lullaby by J. Haydn. Performed on piano. Multi-F0 scores: Precision = 0.93444, recall = 0.49732, accuracy = 0.48055. Note Track scores: Precision = 0.27354, recall = 0.24898, F-measure = 0.26068. An average performance of model BW2 as a lot of false negatives are estimated. The main part of the tones has some parts covered by an estimation, but commonly the offset is estimated too early. Some false positive estimates also seem to occur at 1000 or 2400 Hz.

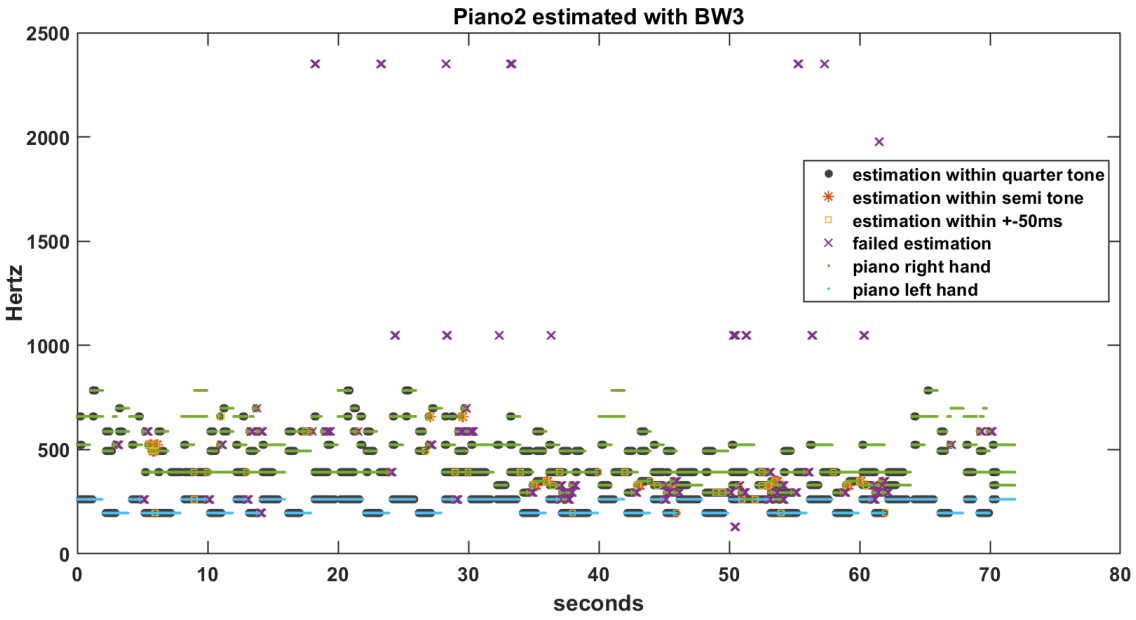

Figure 33: Lullaby by J. Haydn. Performed on piano. Multi-F0 scores: Precision = 0.95114, recall = 0.48539, accuracy = 0.47358. Note Track scores: Precision = 0.23451, recall = 0.21633, F-measure = 0.22505. BW3 has very similar measures and plot as BW2 in Figure 32, even if the parameters are trained on different data.

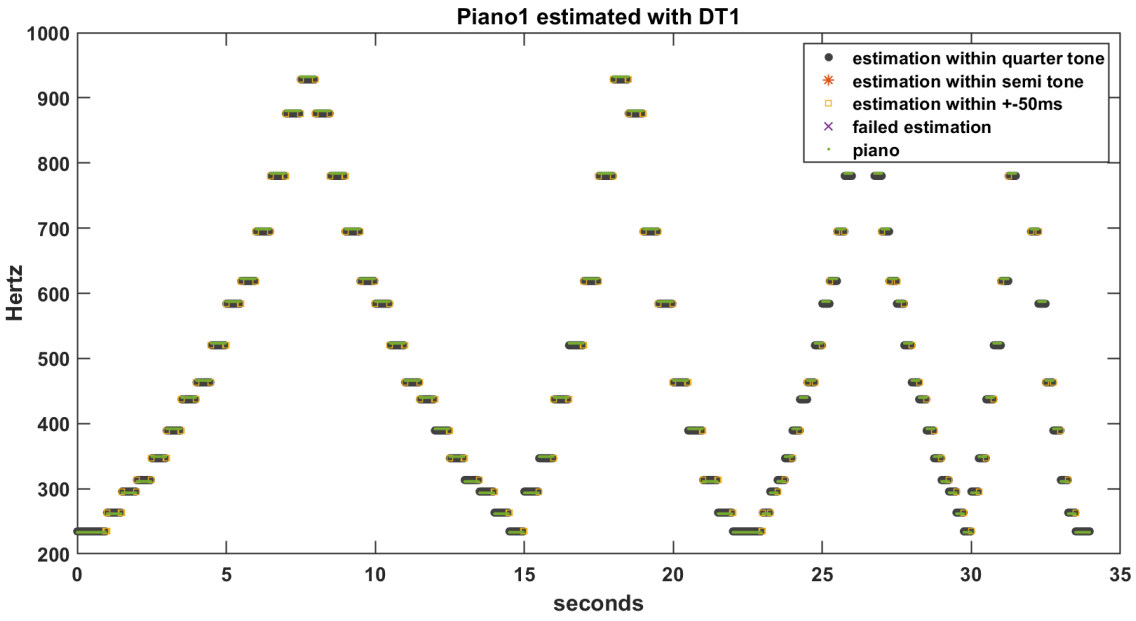

Figure 34: The B flat scale. Performed on piano. Multi-F0 scores: Precision = 0.93444, recall = 0.96189, accuracy = 0.88137. Note Track scores: Precision = 0.98851, recall = 0.98851, F-measure = 0.98851. A simple test file but with an almost perfect estimation. There is one tone where the offset is missed (the F-measure for Note Track without offset is 1), but which tone is impossible to see from the plot.

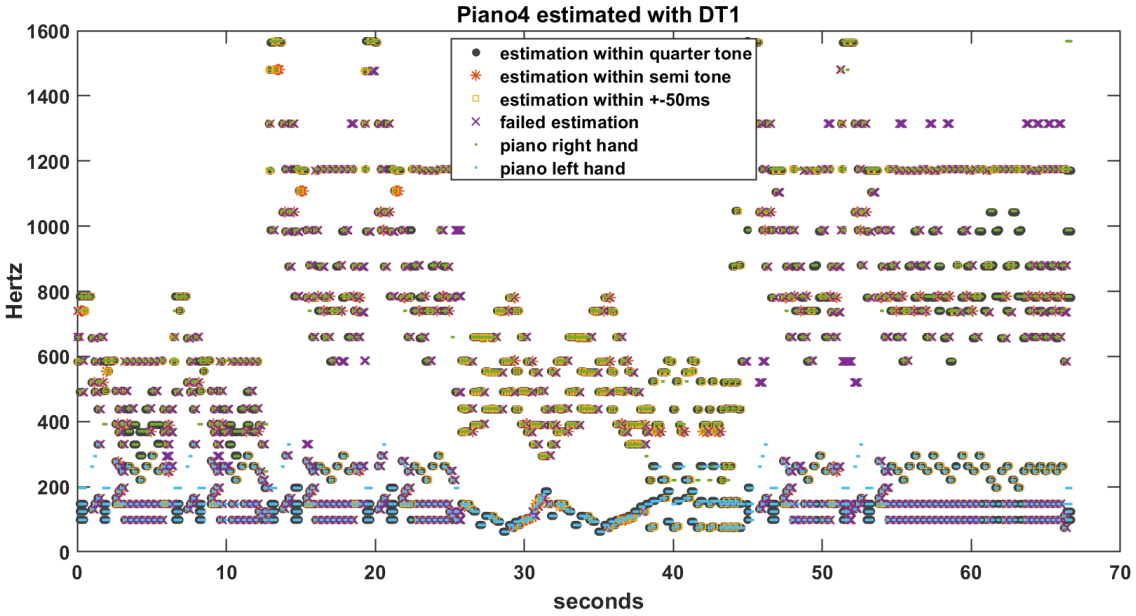

Figure 35: Trepak by P. I. Tchaikovsky. Performed on piano. Multi-F0 scores: Precision = 0.58251, recall = 0.75757, accuracy = 0.49098. Note Track scores: Precision = 0.16349, recall = 0.12794, F-measure = 0.14355. A well performed estimation by DT1 on this complicated test file with many short tones and chords. Few false negatives are estimated, resulting in a high recall. The performance in the high frequencies appears similar as in the lower frequencies. The false positive estimates do mainly appear between two repeating tones. This is seen at 150 Hz from 3-26 and 48-66 seconds where two tones are repeating at two different frequencies. At 150 Hz again but from 26-45 seconds the tones are not repeating and the failed estimates are much fewer.

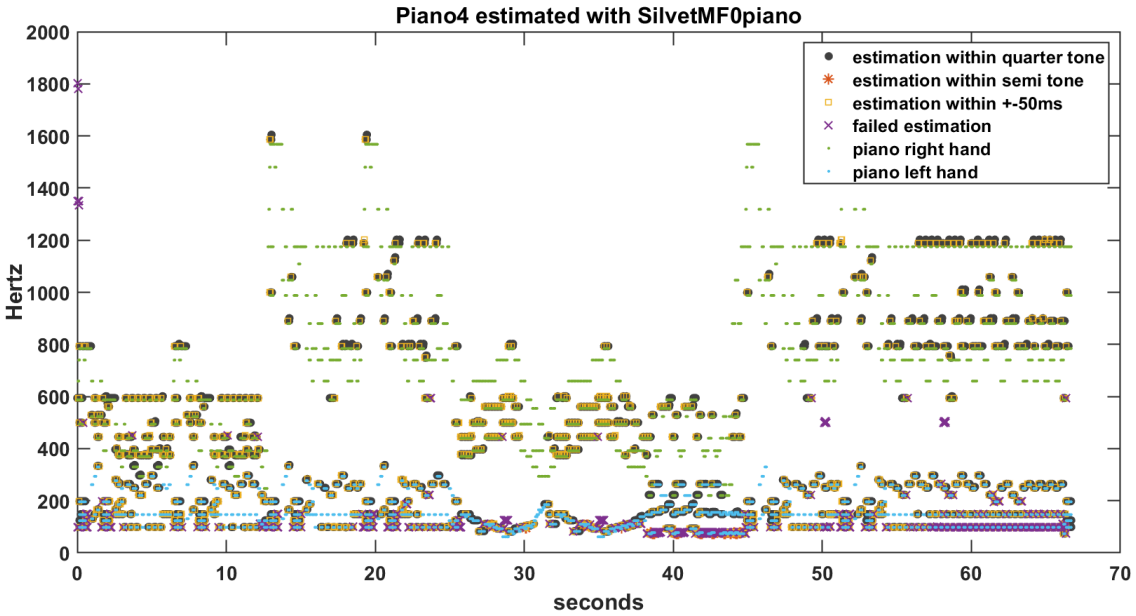

Figure 36: Trepak by P. I. Tchaikovsky. Performed on piano. Multi-F0 scores: Precision = 0.59000, recall = 0.47999, accuracy = 0.35993. Note Track scores: Precision = 0.14879, recall = 0.11538, F-measure = 0.12997. Compared to Figure 35, the same type of error with repeating tones occur, but only in the lower register. In general there are many false negatives estimated resulting in low recall. But there are many true positives, that is the tones which are estimated are in general correct, especially for the frequencies > 200 Hz, resulting in a relatively good precision.

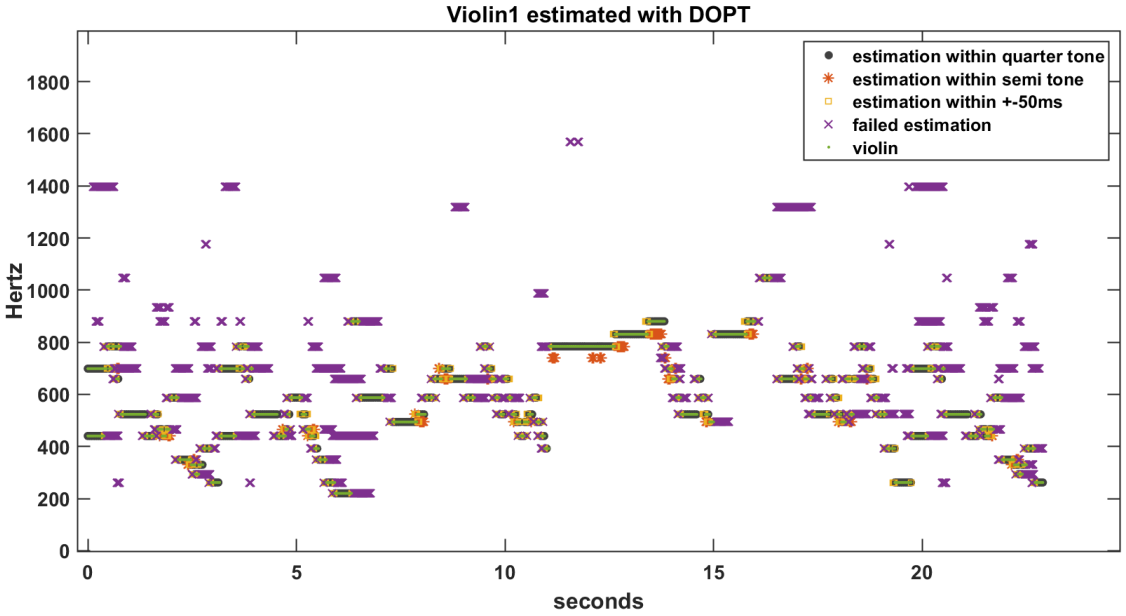

Figure 37: The Divertimento by W. A. Mozart. Performed on violin. Multi-F0 scores: Precision = 0.40795, recall = 0.98256, accuracy = 0.40502. Note Track scores: Precision = 0.05983, recall = 0.10687, F-measure = 0.07671. DOPT does as in Figure 30, estimate few false negatives and receives a high recall score. Differently from the piano2 estimation, the failed estimates appear in the higher frequency region and can commonly be mapped as the first overtone of a ground truth fundamental frequency.

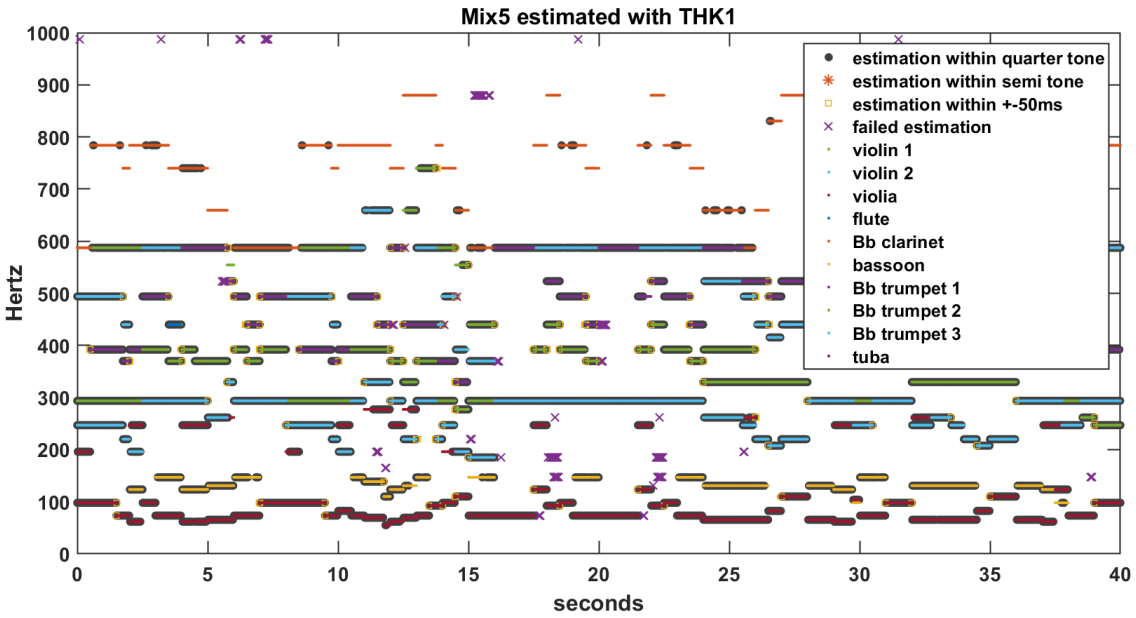

Figure 38: Hark! The Herald Angels sing by F. Mendelssohn. Performed on violin. Multi-F0 scores: Precision = 0.97401, recall = 0.53655, accuracy = 0.52898. Note Track scores: Precision = 0.26357, recall = 0.14011, F-measure = 0.18296. There are lots of tones to keep track of in this test file, but the THK1 estimation performs very well. The only tones which seem problematic is the clarinet at 700-900 Hz where many false negatives are estimated, that means quite few ground truth tones are estimated.

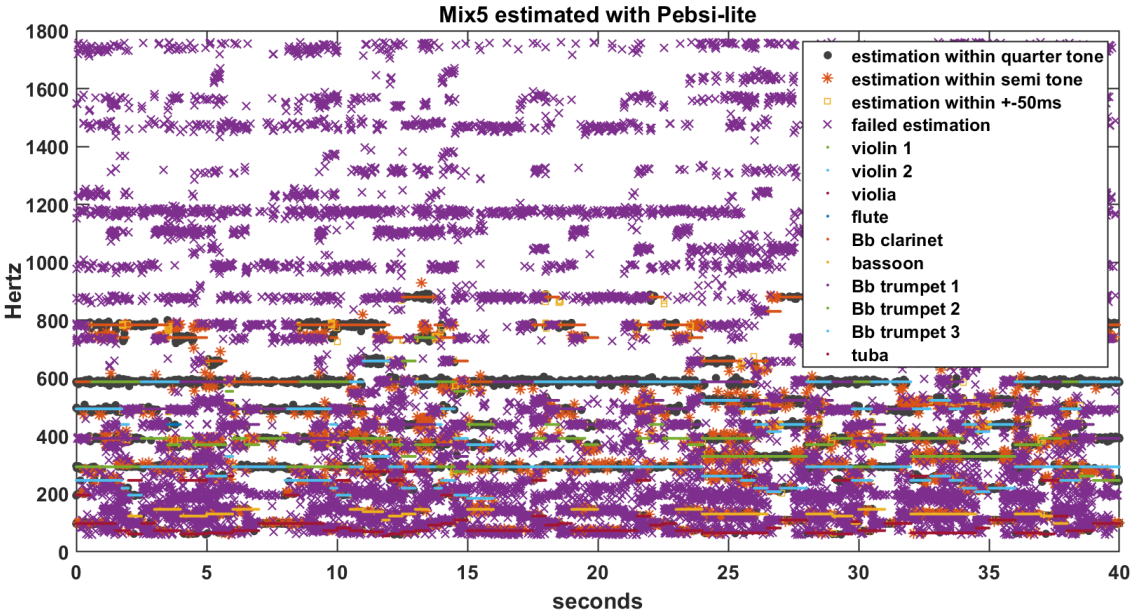

Figure 39: Hark! The Herald Angels sing by F. Mendelssohn. Performed on violin. Multi-F0 scores: Precision = 0.45456, recall = 0.26800, accuracy = 0.20279. Note Track scores: Precision = 0.00000, recall = 0.00000, F-measure = 0.00000. The estimation by PEBSI-Lite appears very noisy with a lot of false positives. Irregardless of that, the precision and recall appear not extremely low for Multi-F0. Study Figure 40 for further analysis.

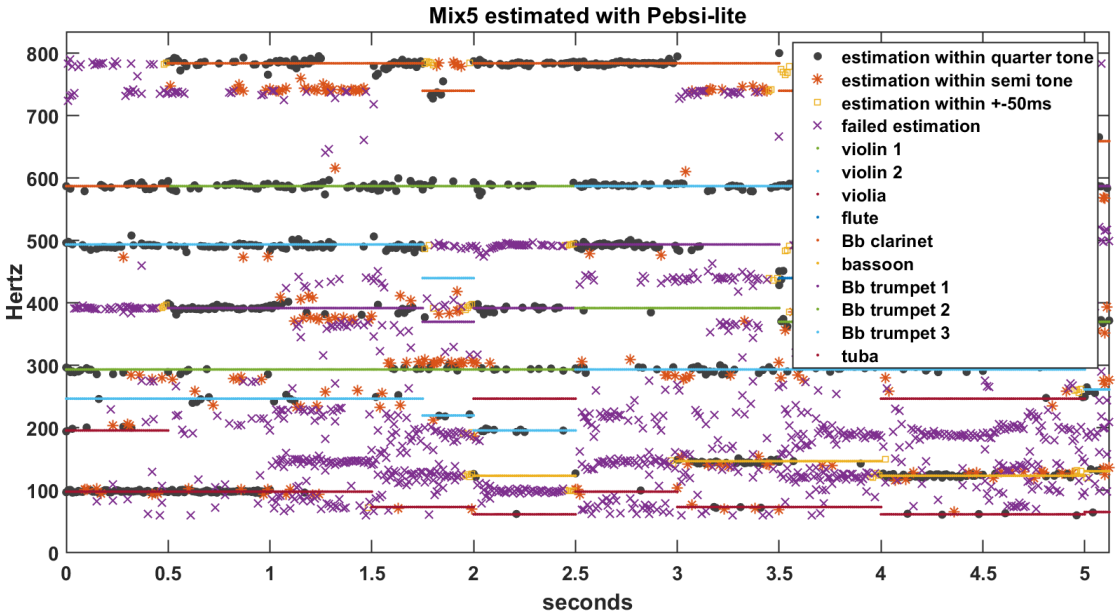

Figure 40: Zoomed in version of the first 5 seconds in Figure 39. About half of the ground truth tones are successfully estimated in each time step. A couple of semitones are estimated, illustrated by an orange stars at 300 Hz from 1.5-2 seconds for example. This error hasn't appeared for the other models in the same extent and can be explained by the high time resolution with lower frequency resolution in the model.

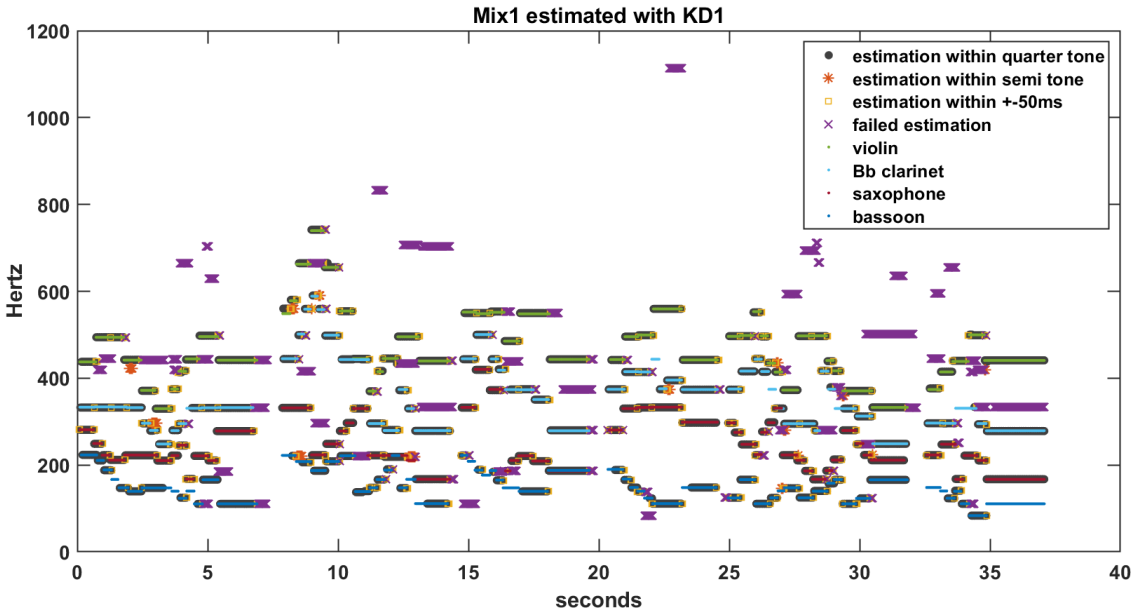

Figure 41: Nun bitten wir den heiligen Geist by J.S. Bach. Performed on violin, clarinet, saxophone and bassoon. Multi-F0 scores: Precision = 0.75322, recall = 0.85980, accuracy = 0.67083. Note Track scores: Precision = 0.43548, recall = 0.46154, F-measure = 0.44813. A good estimation of KD1. The balance between precision and recall is quite good, especially for the Note Track task, resulting in a high accuracy and F-measure. It is noticeable how the failed estimates seem to be perfectly spread over the frequency (and time) axis. Errors can be identified, but no structural errors. Similarities has only been identified for neural network models with a very low error rate.

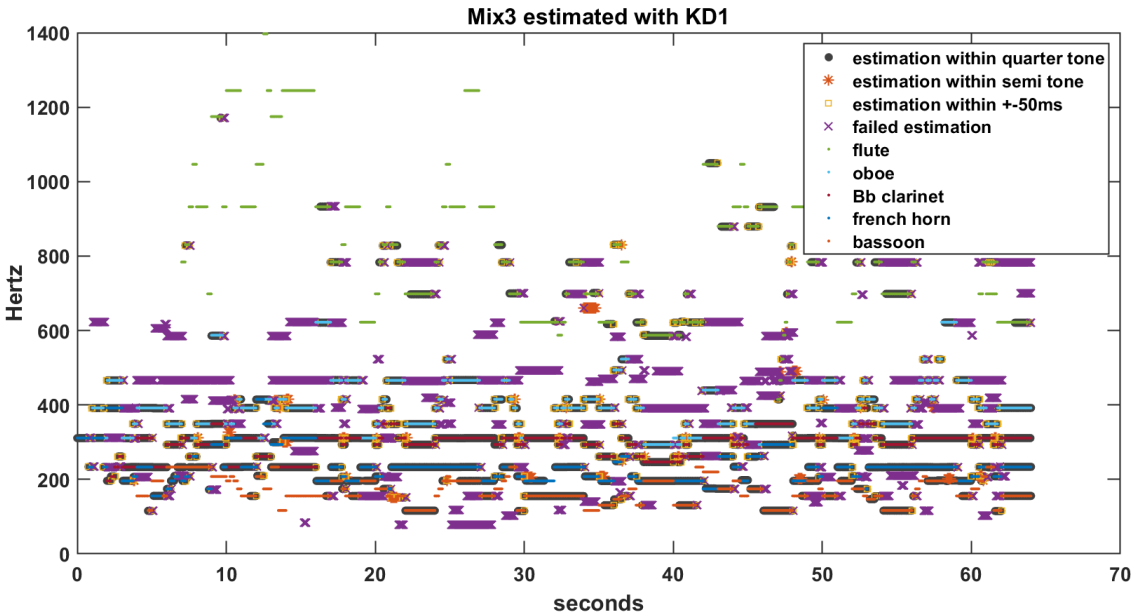

Figure 42: Tochter Zion by G. F. Händel. Performed on flute, oboe, B^b-clarinet, french horn, bassoon. Multi-F0 scores: Precision = 0.60421, recall = 0.73074, accuracy = 0.49418. Note Track scores: Precision = 0.11211, recall = 0.10482, F-measure = 0.10834. Compared to Figure 41, there are lots of structural errors such as the flute is estimated as false negatives, that is the true tones of the flute is not estimated. Also the tones of the oboe are estimated to be much longer than the ground truth. The french horn and the clarinet in the middle frequency register seem to be the only instruments which are well estimated.

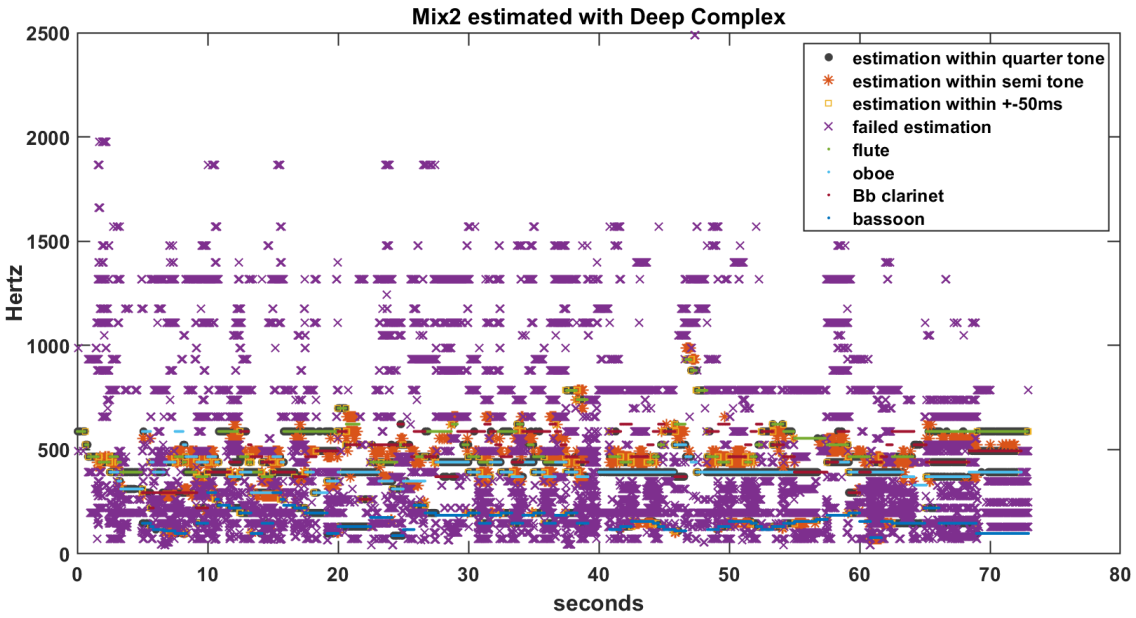

Figure 43: "Graduale" from th Requiem by L. Cherubini. Performed on flute, oboe, clarinet, bassoon. Multi-F0 scores: Precision = 0.30622, recall = 0.52641, accuracy = 0.24008. Note Track scores: Precision = 0.00493, recall = 0.08333, F-measure = 0.00930. A quite bad estimation by the Deep Complex model with lots of false positives and lots of false negatives. It is a bit noisy to see, but the bassoon is in a large extent missed.

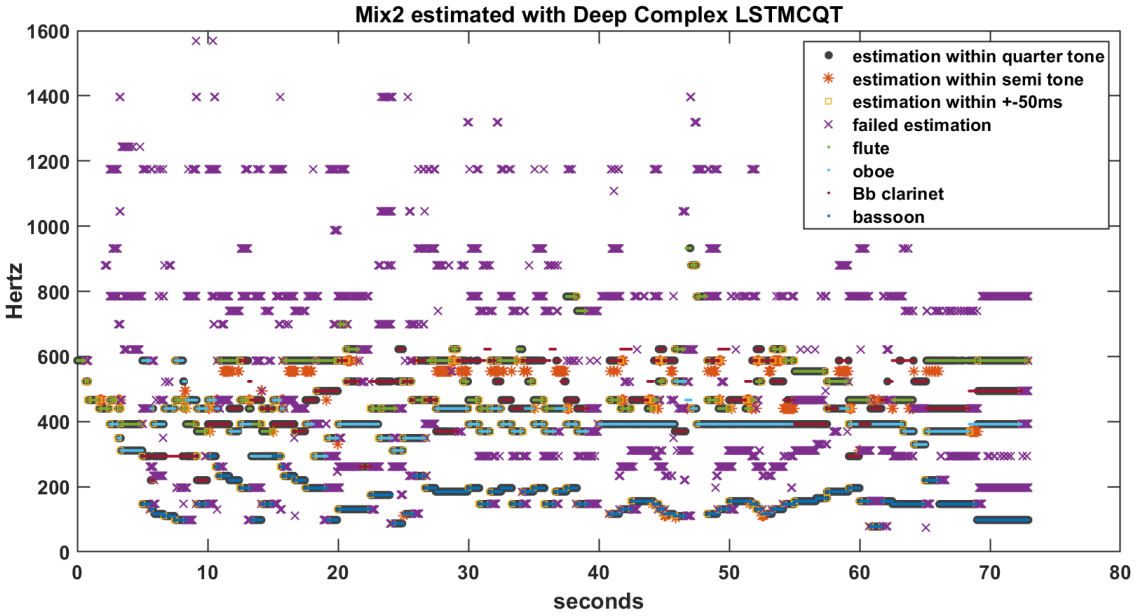

Figure 44: "Graduale" from th Requiem by L. Cherubini. Performed on flute, oboe, clarinet, bassoon. Multi-F0 scores: Precision = 0.67075, recall = 0.80831, accuracy = 0.57870. Note Track scores: Precision = 0.05652, recall = 0.29333, F-measure = 0.09478. Compared to the 43, there are much less false positives. The failed estimates which occurs can often be mapped as the first overtone of the true fundamental frequencies, for example at 300 Hz for 30-75 seconds, the ground truth bassoon has similar frequency movements on half the frequencies. Similarly at 800 Hz, which has similar frequency movements as the oboe at 400 Hz.

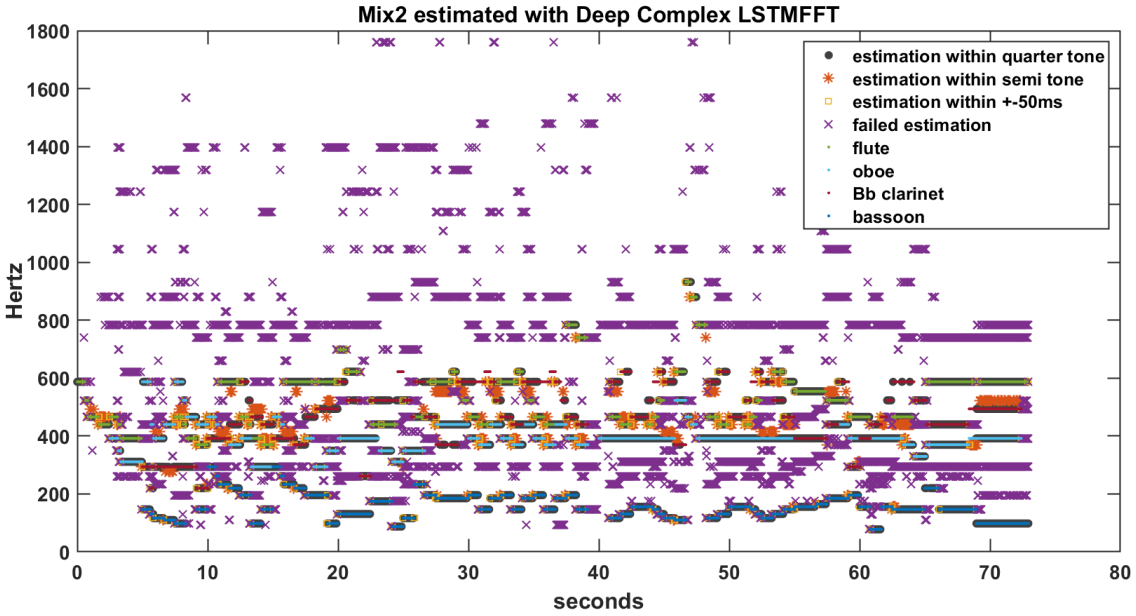

Figure 45: "Graduale" from th Requiem by L. Cherubini. Performed on flute, oboe, clarinet, bassoon. Multi-F0 scores: Precision = 0.52875, recall = 0.81376, accuracy = 0.47167. Note Track scores: Precision = 0.00571, recall = 0.04667, F-measure = 0.01017. Similarly to 44, the LSTMFFT model has few false negatives and high recall. It also has many false positives at the first overtone of the bassoon and the oboe. One may also find semitones to be estimated at for example 600 Hz from 70-75 seconds and they appear next to a correctly estimated pitch by the clarinet. The error source could either be a bad spectral representation where two adjacent frequency bins shares the power in a fundamental frequency which appears between them, or that the convolutional layer convolves the spectral power in a non-optimal way. Similar issues can be seen in the LSTMCQT model as well.

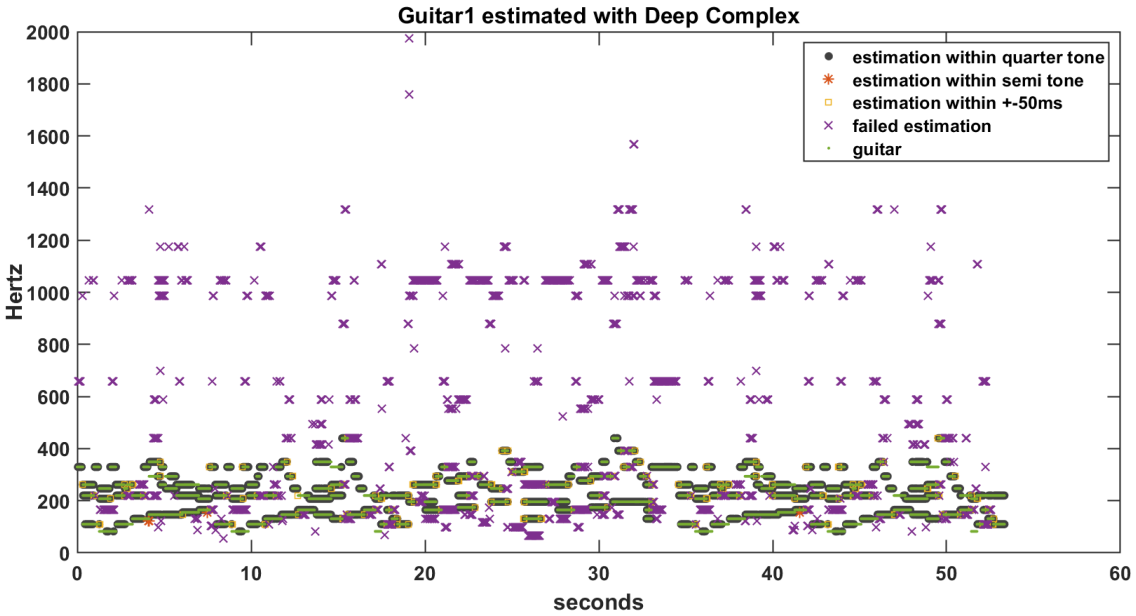

Figure 46: Adantino by F. Carulli. Performed on guitar. Multi-F0 scores: Precision = 0.72149, recall = 0.75169, accuracy = 0.58263. Note Track scores: Precision = 0.06275, recall = 0.27391, F-measure = 0.10211. The Deep Complex model seems to perform quite well, as all ground truth tones looks to be estimated. But considering the recall score for Multi-F0, it is significantly less than both Figure 47 and 48. That yields that many false negatives are estimated, for some time steps which cannot be seen in the plot.

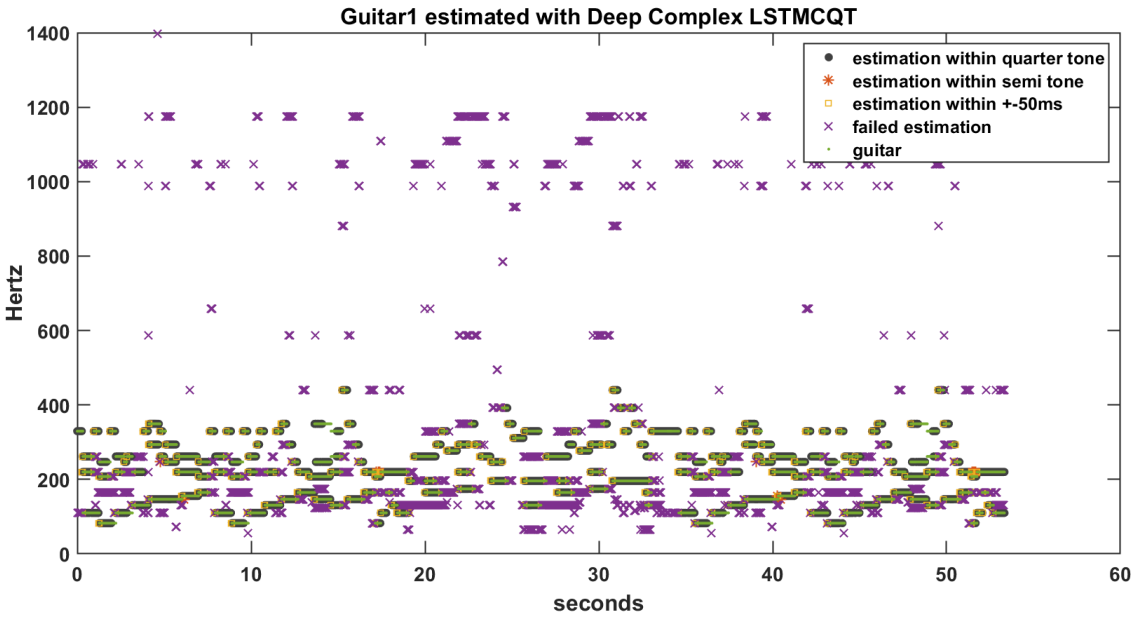

Figure 47: Adantino by F. Carulli. Performed on guitar. Multi-F0 scores: Precision = 0.67649, recall = 0.87181, accuracy = 0.61529. Note Track scores: Precision = 0.09581, recall = 0.37826, F-measure = 0.15290. The LSTMCQT performs slightly better than the Deep Complex model does. The same failed estimates appear at around 1000 Hz but also some more in connection to a ground truth tone where the onset or offset is missed.

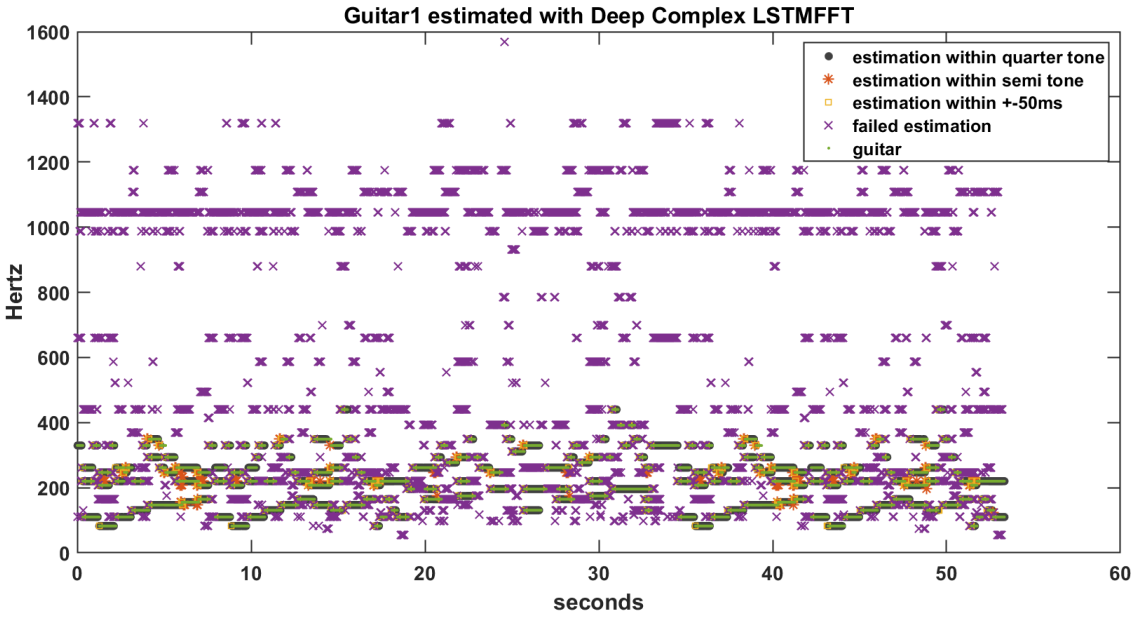

Figure 48: Adantino by F. Carulli. Performed on guitar. Multi-F0 scores: Precision = 0.45502, recall = 0.86302, accuracy = 0.42437. Note Track scores: Precision = 0.00232, recall = 0.01739, F-measure = 0.00410. The similar structures as could be seen in the Figure 46 and 47. Similar recall as the LSTMCQT model but lower precision, resulting in the lowest accuracy of the Deep Complex models.

7.4 Analysis of Results

First, the average scores for the Multi-F0 and Note Track tasks will be considered for analysis, and the best performing models will be discussed. Thereafter, some interesting models or specific test files will be considered.

The model THK1 can be concluded to perform significantly better than all the other models in the Multi-F0 task, since it achieves the highest averaged accuracy scores in Figure 16[72]. The model CT1 clearly has the 2nd best accuracy, followed by the herein proposed model Deep Complex LSTMCQT. The significance for the THK1 model is clear both for the ensemble test files and the piano test files. Considering each individual test file in Section 7.2.2, one may note that the models THK1 and CT1 are very robust and achieves high accuracy scores for all types of files, while for example the models MM1 and DOPT performs better on piano music than ensemble music. THK1 has more equal scores of precision and recall than the rest of the models, while CT1 has a higher precision than recall for the Multi-F0 task.

Furthermore, considering the Note Track task, average scores are presented in Figure 20. CT1 performs significantly better than the other models and achieves the highest F-measure for both the ensemble test files and the piano test files with a margin. The models THK1 and DT1 perform significantly better than the fourth best model. Again the models DOPT and MM1 perform well on the piano test files but worse on the rest. The proposed model performs relatively bad on the Note Track task compared to the Multi-F0 task. The CT1 model is found to have a more equal score of precision and recall which was not the case for the Multi-F0 task. Considering Note Track with no offset criterion in Figure 24, CT1 might be one of the models which F-measure increases the least, but it is still the best model considering all test files. For the averaged piano test files, the models DOPT and MM1 performs better and achieves shared 2nd best accuracy with DT1.

Overall it can be said that the THK1 and CT1 are the best performing models in the evaluations. The models' estimates of the test file strings1 are plotted in Figure 25 and 26. Both models perform very well on this test file and receives the two highest accuracy scores and F-measure scores in both the Multi-F0 and the Note Track task. The models are performing very similarly and succeed to estimate both high and low frequencies very well. Both models also have problems with the octave jumps of the violincello as described in the figure texts. One thing that can be seen is that the CT1 model estimates more false negatives than THK1, resulting in a lower recall score for the Multi-F0[73]. But for Note Track, the recall score is higher for CT1. This is perhaps not obvious from the plots, but THK1 seems to be trained to find as many frequencies as possible, while CT1 seems to be trained to find as many complete tones as possible. One may see in the plots that THK1 has more pink crosses that are sparsely spread, while CT1 obtains such errors in clusters. In Note Track, a cluster of failed estimates are as bad as a lonely failed estimation, while each time bin for a failed estimation counts equally in the Multi-F0 task. Also, in Note Track, a couple of correctly estimated frequencies may count as failed estimates if the length of the estimated tone is not enough. The different models are thus optimized for different purposes and may therefore achieve different scores in the two tasks, even if the plots indicate similar performance.

[72] No tests has been made to statistically prove that the results are significant. The term significant indicates a large difference, at least $> 5\%$ better than the second best model.

[73] For specification of TP, FP, TN, and FN, see Section 4.2

In order to explain how well THK1 and CT1 are performing compared to other models, the estimation of MHMTM1 on the test file strings1 is plotted in Figure 27. MHMTM1 performs relatively well on strings1, achieving the 4th best accuracy score in the Multi-F0 task. Structural errors can moreover be found, such as overestimating lower frequencies, i.e., a lot of false positives, giving a high recall but low precision in the region of frequencies below 200 Hz. On the other hand, many ground truth frequencies above 400 Hz tend to not be estimated by the model, i.e., many false negatives, which results in high precision but low recall. In order to optimize the accuracy, the precision and recall should be similar, which is clearly not the case. This is an issue that doesn't appear in THK1 and CT1, which indicates a structural advantage for those models.

Another impressive result for THK1 and CT1 is shown in Figure 28 and 29, for an advanced piano test file with many fast and short tones. From the plots it can be seen that almost all true pitches are estimated, i.e., with few false negatives. However, many yellow squares appear, indicating that the estimates are slightly off in the time domain but correct in the frequency domain. This error does only affect the Multi-F0 score negatively. That is probably the reason why the Note Track scores appear higher than the Multi-F0 scores. Interestingly, the only obviously error in the estimates is made in the end, where six long tones are played. One could otherwise guess that long tones would be easier to estimate since a high time resolution is not needed. Probably the uncommonly high tone at 1900 Hz is confusing the models.

Two other models that are performing well and worth discussing further are DOPT and MM1. The models achieve high scores for all piano test files, Figure 18 and 22. They also have the second best F-measures for Note Track without offset criterion considering the piano test files, Figure 24. For almost all other test files, these models perform averagely. This may be explained by the fact that the models are built on neural networks trained on piano music only. The models seem to learn very well how to estimate piano music, but no other instruments. The estimates on piano2 for the two models are shown in Figure 30 and 31. The two models perform similarly in the accuracy and F-measure, but quite differently when it comes to precision and recall. DOPT tends to get high recall and lower precision, and MM1 tends to get high precision and lower recall. From the plots this becomes clear, as DOPT has almost no false negatives but has plenty of false positives. The MM1 model has very few false positives but quite many false negatives. One may also note that both models are very accurate in finding the onsets. However, for the offset, a structural error seems to appear for the two models, where MM1 tends to put the offset too early while DOPT tends to put the offset too late. Another structural error seems to be that DOPT mistakenly estimates low frequencies while MM1 mistakenly estimates high frequencies.

The results of MM1 are also interesting since it performs better than DT1 in MIREX 2016 for the Multi-F0 task on ensemble music and DT1 performed better than MM1 on Note Track on piano music. The comparisons herein show that DT1 scores higher in both Multi-F0 and Note Track. Since DT1 is trained on both ensemble and piano music, these new Multi-F0 results are a more expected. An interesting comment about the DOPT and MM1 models is their performance on the guitar1 test file. In the Note Track task, see Figure 23, it can be seen that they may perform well for other instruments than piano. This can be explained by the fact that the sound of a guitar and a piano are similar as they both are stringed. Also their overtone structures are very similar, as explained in Section 3.1.1.

Considering another test file, the plot of estimates by BW2 and BW3 on piano2 is illustrated in Figure 32 and 33. In this case BW2 is trained on ensemble and piano music while BW3 is trained on piano music only. Both models are trained for Note Tracking. Interestingly, the model trained on ensemble and piano music gets a higher score in both accuracy and F-measure for the piano2 test file. By comparing the plots of the estimates, one notices that they are very similar. They almost have the same tones estimated correctly and incorrectly and the small differences which occurs may just as well be random. By studying the Note Track results for all the piano files, Figure 22, one may note that BW2 and BW3 tend to perform similarly, while in Figure 21 BW2 seems to perform better than BW3. One may question the idea of focusing a BW model on piano music by training it on piano music only, since that didn't seem to improve the model. This is contrary to the conclusion about DOPT and MM1 where their high performance on the piano test files was explained by the fact that they were trained on piano music[74]. One difference between DOPT & MM1 and BW2 & BW3 is that DOPT and MM1 are neural network models while BW2 and BW3 are probabilistic models. The probabilistic models seem to be more robust to the type of training data. This is confirmed by studying SilvetMF0 and SilvetMF0piano in Figure 17, where the model trained on piano performs almost as well as the model trained on ensemble and piano music.

By this discussion, the training data seems extra important for the neural network models. Those models trained on the MAPS dataset or MIDI files have a natural advantage in the tests herein since the main part of the test files comes from MIDI files. If the best performing models would have been trained on MIDI files, the results of this evaluation could be questioned. Perhaps DT1 would not achieve the third best F-measure in the Note Track task if it hadn't been trained on MIDI files. The THK1 and Deep Complex LSTMCQT models are trained on MusicNet which is a real recorded music dataset. Unfortunately, the dataset which CT1 is trained by is not public or detailed. MusicNet may be considered a very powerful dataset, since it can be used to outperform models trained on MIDI files for test files constructed by MIDI files. Interestingly, the network architecture of THK1 is quite shallow which might indicate that a music transcription model is more dependent of a good training dataset than adding more advanced architecture of the network or more feature detections. On the other hand, CT1 has more advanced architecture and feature detections. Whether a perfectly working model is achieved by improving the training dataset or by changing the architecture and adding more features, or both, is an unresolved in this thesis. This question should be further investigated.

The test file piano1 is quite special and was included in the test set to be the simplest possible melody to estimate. It consists of a single pitch melody, or more specifically a chromatic scale, on a piano. Since the models are able to estimate quite advanced polyphonic melodies, this simple melody was expected to achieve full score of almost all models. However, most models actually struggled with it, as can be seen in Figure 18 and 22. One model lived up to the expectations and achieved an almost full score in the Note Track task. That was model DT1 and its corresponding estimates for piano1 are shown in Figure 34. A couple of yellow squares in the plot indicates small misses in the time dimension, reducing the Multi-F0 score but not the Note Track score. The Multi-F0 score is interesting in the sense that a perfectly estimated melody according to the Note Track accuracy, only receives a Multi-F0 accuracy of 0.88137. The source of the errors

[74]While many other models which used ensemble music as training data did not perform as well on the piano test files.

which occur according to the Multi-F0 score is tracked as misses in the onset and offset. If 90% of a half second long true tone (500 ms or 50 timebins) is estimated, the onset and offset may together differ by 5 timebins (50 ms). It is hard to achieve better precision than that according to reasons mentioned in Section 3.1.4. This indicates a limit of what a Multi-F0 model possibly can achieve[75]. One could be disappointed in the performance of the models in general on piano1, but they are not trained for this kind of data. There already exists fine working single pitch estimators for this problem, which are not considered in this thesis.

DT1 scored almost perfectly on piano1, the simplest test file considered. For a more advanced melody, the model struggled, as may be seen in Figure 35 where the estimate for piano4 is presented. DT1 succeeds in finding the main part of the true tones, i.e., few false negatives, both in the high and low frequency register. But a lot of false positive estimates appear at repeating tones, the reason is partly due to the very short tones in the test file. For model SilvetMF0piano in Figure 36, the issue of repeating tones is not salient, which indicates that it is possible to circumvent the problem. The model SilvetMF0piano has other issues, mainly the high amount of false negatives.

A test file similar to piano1, but slightly more difficulty is violin1, which is almost a single pitch melody throughout. The challenge comes from the instrument, since the violin has a large amount of strong overtones while the higher overtones on a piano generally taper off quickly. The models succeed quite well with new false negatives, resulting in a high recall, seen in Figure 19. The problem is that the overtones of the violin are often estimated as fundamental frequencies, resulting in many false positives and low precision. One example of this is illustrated in Figure 37 where the issue is clearly seen along with the offset being estimated too late. One could maybe expect the pitch to be more complicated to estimate due to the vibrating pitch of a violin, but this was not found in the estimates.

The most complicated test file to estimate is mix5, consisting of an ensemble of 10 instruments. This file was included to give the models a challenge, which it turned out to be. Even if the scores in Figure 17 were not very impressive, the plot of THK1 in Figure 38 look quite satisfying. In the figure, almost all tones are perfectly estimated with few false negatives, except the clarinet in the high frequencies. The clarinet is notoriously complicated to estimate due to its overtone structure where the odd numbered overtones are strong and the even are weak. The reason why the accuracy score is so low is that multiple instruments are playing the same pitch simultaneously. The model estimates only a single fundamental frequency, which in the calculation of the measures, only can be set as one instrument. As seen in Figure 17, the precision is high for many models but the recall is often low, which indicates similar issues for several models. The performances on this complicated test file are above expectation by considering the plots of the estimates.

A model that is also worth to mention is PEBSI-Lite, which in this evaluation doesn't seem to perform very well. The model is quite different from the others as it estimates frequencies using only 10 ms data at each step, independently of the adjacent time step estimates. This is the reason why the Note Track scores are close to 0 for almost all test files. Also, the model outputs pitch estimates which are not rounded to the closest musical note, which the other models do. A visualization of the PEBSI-Lite is presented in Figure 39. The plot may look very

[75]For example model THK1 can therefore be claimed to estimate the piano2 test file in Figure 18 perfectly.

nasty as there is a very large amount of false positives, but considering a zoomed version of the first 5 seconds, as in Figure 40, the model doesn't seem to perform catastrophically bad. There are a lot of false positives, resulting in a bad accuracy, but it has a precision and recall which is about half of the THK1 model, which in turn was considered as a very well performing model[76].

A model which was expected to perform a little bit better was KD1. In MIREX 2017 it achieved the second place both in the Multi-F0 task and the Note Track task for ensemble music. But for the tests herein, the model is quite far from the top performing models. There is one test file it performs much better on, namely mix1. A plot of the KD1 estimate of mix1 is shown in Figure 41. In the plot the instruments in the middle register, the saxophone and the clarinet, are ideally estimated. At the same time, a couple of true tones from the bassoon are missed and the offset of the violin quite often appears too late, as well as a couple of false positives in the high and middle register. But in general KD1 performs very well on this test file, unlike the rest of the test files. The reason is probably that the model is using parameters trained on the MIREX development set, which is a real recorded music file just like mix1. Studying Figure 17 and 21, the KD1 estimates for the other ensemble files, which consists of simulated sound, achieves lower scores for the than mix1. In Figure 42, the KD1 estimate of mix3, which is a test file consisting of simulated sound, is plotted. The estimate appears distinctly different to the estimate of the real recorded music test file mix1. Lots of false positive estimates appears where the failed estimates form long continuous tones. Also big parts of the flute appear as false negatives. Thus, the training data seems to be crucial for what type of music the KD1 model succeeds to estimate.

Other models which perform much better on the real recorded music test file mix1 than the rest, are BW1, BW2, and BW3. On both the Multi-F0 and the Note Track task, these models achieve the highest accuracy and F-measure after THK1 and CT1 respectively for mix1, Figure 17 and 21. Both BW1, BW2, BW3, and KD1 were performing very well in MIREX, but they seem to be focused on only the type of data which appears in MIREX, real recorded music. The test files generated from MIDI files in this thesis seem to not be in favor of these models. Considering a real life application of a music transcription model, one would almost always consider real recorded music, which they work quite well in. Neither BW1, BW2, BW3, or KD1 are neural network-based models. Also, the parametric model PR performs better on the real sound test file than the simulated sound test files. From Figure 17 and 21, the neural network model THK1, trained on real sound, and DT1, trained on MIDI files, both seem to be able to deal with both simulated and real sound, independently of what kind of data the model has been trained on. This is a sign of robustness for the neural network models. There are exceptions of non-neural network models which may handle both types of data, such as the MPE models, Figure 17, but the trend is quite clear that the neural network models is better of dealing with different types of data.

A general observation from the models' performances is that false positive estimates are most common in either very high frequencies, far from the ground truth tones, or at the end of the tones where the offset is estimated too late. The false positives which occur far from the ground truth are usually octave errors or other multiples of a fundamental frequency, where an overtone is mistakenly estimated as a fundamental frequency. This is probably the most common

[76]Some of the failed estimates by PEBSI-Lite are semitones which are estimated due to the high time resolution and low frequency resolution. Exclusively most false positives can be linked as overtones which mistakenly are estimated as fundamental frequencies.

error. The failed estimates corresponding to the missed offsets were expected, due to issues with the offset determination described in Section 3.1.4. False positive errors which don't occur as commonly as expected are estimates occurring between a quartertone and a semitone, which would occur due to bad frequency resolution. The errors which are indicated as such in the plots are mainly due to late offsets and that the following ground truth tone jump is a semitone. The reason why this type of error doesn't appear that frequently is possibly due to the perfectly tuned simulated instrument sounds from the MIDI files or that the models are using well working spectral representations.

Finally the Deep Complex models will be discussed. The models are designed for the Multi-F0 task, which will be in focus. First, considering Figure 16, it can initially be noted that the Deep Complex LSTMCQT achieves the 3rd highest accuracy based on the average of all models. This is a very pleasing result. Comparing the different Deep Complex models one may note that the Deep Complex LSTMCQT has a slightly higher average accuracy than Deep Complex which in turn has slightly higher than the Deep Complex LSTMFFT. The proportions are almost equal considering only the averaged ensemble or piano test files. This suggests a structural improvement of the LSTMCQT model and a structural degradation of the LSTMFFT model compared to the Deep Complex model. The three models are very similar in their structure, but the few differences seem to have a quite large impact. The small differences in the model structures give quite large differences in the estimates, which is illustrated in Figure 43, 44, and 45, where the estimates of the test file mix2 are shown. For this test file, the Deep Complex model performs poorly and a large amount of false positive estimates are apparent both in the high and the low frequency region. The estimates from the LSTMCQT model are cleaner as there are not as many false positive estimates even if there clearly are some. However, the number of false negatives is low. The false positives seem to appear at frequencies which can be linked as overtones of the true fundamental frequencies. For example, the false positives around 800 Hz have a similar form as the oboe playing around 400 Hz. The same is true for the false positives around 300 Hz between 30-75 seconds, where a similar form as the tones from the bassoon appear around 150 Hz. The plot of the LSTMFFT model have similar look as the LSTMCQT where the overtones seem to be mistakenly estimated as fundamental frequencies. The LSTMFFT also has more false positives than LSTMCQT. The optimized threshold for the LSTMFFT model was a bit lower than the LSTMCQT, which might be a reason why its number of false positives is higher.

Also, the Deep Complex models are considered for the test file guitar1 in Figure 46, 47, and 48. For this test file the standard Deep Complex model performs better and achieving the highest precision of the three models while the LSTMCQT and LSTMFFT models achieve a higher recall. The reason is probably that the estimates of the Deep Complex model have limited relation to the adjacent time steps. Studying Figure 46, it looks like almost all true frequencies are estimated, but from the recall score it says that about 25% was false negative estimates. Since the Deep Complex model estimates each time bin independently, it fails to estimate the frequency for all time bins of the tone. A disadvantage of using a relation of the adjacent time bins can be seen for the LSTMCQT and LSTMFFT in Figure 47 and 48 where some tones are estimated to be longer than the ground truth, i.e., the offset is estimated to late. Another issue is that a false positive estimate may affect the following estimates to also become a false positive estimation. This makes the false positives of the Deep Complex model to last for shorter periods than the false positives do for LSTMCQT and LSTMFFT. This is most clearly seen for the LSTMFFT. In general, the Deep Complex model achieves a precision close to the Deep Complex LSTMCQT,

but a lower recall resulting in an overall better accuracy for the LSTMCQT model.

Finally it needs to be stated that it is much easier to point out structural errors than it is to solve them. The music transcription research is based on both scientific discoveries and empirical tests. Thus, it is not always simple to point out exactly what is needed to correct an error. The best approach is probably to add a feature which could correct the error and then empirically test if it worked. The goal of this thesis is not to solve the structural errors which occurs in the different models. Instead this thesis reveals what can be improved and presents differences in the approaches of the models wherein the solution might exist.

8 Conclusions and Summary

The problem formulations in the outset of the thesis were: How well does the best current music transcription model performs. How does it work and why does it work so well? Can complex-valued neural networks enrich the music transcription research area? These questions will be answered in this Section 8.1-8.3 followed by some further elaborations in Section 8.4.

8.1 Summary of the Test Results

The results on the test files show that there is one model, THK1, which is clearly superior in Multiple fundamental frequency estimation, and that there is another model, CT1, which is clearly superior in Note Tracking. This is the same result achieved in MIREX 2017, which strengthens the credibility of both the evaluation made in this thesis and the MIREX evaluation. This thesis also illustrates how well the models perform by visualizing the estimates against the ground truth for different models. Even if the accuracy score was not perfect for the best models, the plots of the estimates showed a satisfying result where only minor structural errors could be found. This indicates that the respective researchers have come far in the process towards automatic music transcription. But even if the structural errors barely exist, there are things which may be fine tuned in order to make the models even more accurate, perhaps the missing component is already existing as a part of another model.

8.2 Discussion of Model Features

To find the most valuable features it is natural to start studying the two best performing models, THK1 and CT1. Both models are neural network based models using CNN. Both models use a logarithmically spectral representation, one using the sliced CQT and one using hand-crafted filter banks. The outputs of the networks are then postprocessed to output the final estimates, one uses a linear classifier and one uses estimated features such as the onset and offset. The models are concluded to be quite similar, but there are a couple of differences.

In THK1, much effort is put into training the network such that no overfitting should occur. This is very reasonable since for example the spectrum of the piano tone E3 activates different parameters in the network than the piano tone F3, but the spectrums of the two tones are very similar, just a shift in the frequency. Randomly shifting the training data makes the model learn the structure of a piano tone and not only the structure of the piano tone E3. The THK1 model has a surprisingly shallow neural network model, only consisting of one convolutional layer and one fully connected layer. This indicates that the deeper models are perhaps too complicated to be trained correctly with the amount of available data. Finally, it should be mentioned that the model is trained on the new dataset MusicNet, which can be motivated to be the best one available, due to its size and all its combinations of instruments.

CT1 has a more advanced and deeper network structure. It uses a succession of convolutional layers and a convolutional LSTM layer in two different channels where one channel is estimating the onsets and is used as input to the pitch estimation channel. The onset approach is motivated by its appearance in other successful models, such as MM1 and DOPT. These models were solely trained on piano data, which makes them perform well on the piano test files only. THK1 includes temporal dependence, but only for 1/3 of a second. Due to the convolutional LSTM layer in CT1, the model may use longer time dependencies as well. It is not clear from the

results herein whether a longer time dependence is important. Some time dependence between the estimates can nevertheless be motivated by the weaker performance of the Deep Complex model and PEBSI-Lite, which don't use any time dependence at all.

By considering the average results of both Multi-F0 and Note Track, Figure 16 and 20, at least the three best performing models are neural network based. This clearly indicates that neural networks are the state-of-the-art in music transcription. One could argue that all the parametric and probabilistic models should be forgotten and all focus for further research should be on optimizing the architecture of the neural networks, but there are features used in the old models which could probably also be used in the neural network models. One example is postprocessing, where the output of the neural network is very similar to many of the probabilistic and parametric pitch estimators, in which the postprocessing is usually more advanced than a binary threshold, which is used in some neural network models. Another example is the onset detector, which is already used in a couple of neural network models and seems to give effective results. But other feature detectors could probably also be added to the networks such as beat or chord estimator. It can be argued that feeding as much valuable information for music transcription as possible into the neural network, only improves the network. The network could then be able to learn by itself in what extent these features should be used.

Herein, complex-valued neural networks are proposed. It has been shown in this thesis that a complex-valued neural network, even if the structure is very simple, can match the best performing real-valued models. It is true that the Deep Complex LSTMCQT model was quite far from both THK1 and CT1, but considering the simple structure of the model consisting of a neural network and a binary threshold, the result is quite impressive. One part of its success is probably that it has been trained on a solid dataset, and compared to THK1, which is the only model which has been trained on the same dataset, it performs badly[77]. A couple of potential improvements of the Deep Complex model are suggested in Section 8.3. The use of complex values in neural networks for music transcription is theoretically motivated by that a neural network performs better if it is given more usable information. The complex-valued representation of a signal which keeps both phase and amplitude intact, is herein claimed to be more useful than only the absolute value.

The THK1 and CT1 may be the best performing models right now, but the research is changing fast. It can be expected that the few structural errors identified herein being overcome in a few years, perhaps using some of the improvements suggested herein. Right now one might be able to fine-tune the estimates to look perfectly in a plot. Further, the goal will be to achieve the perfect precision and recall scores[78].

[77] Almost the same dataset. THK1 add some actions to avoid overfitting to the dataset which are not considered for Deep Complex LSMTCQT.

[78] An example of a research field which has come a step further than the music transcription field, is the research about hand written digits. Models for the MNIST dataset are optimized to achieve the perfect precision. Currently, year 2018, the error is 0.21%. It should be noted that this level of fine tuning not is possible in the music transcription field at the moment, for example due to an approximate 4% error in the annotation of the MusicNet training dataset.

8.3 Analysis of the Proposed Models

The herein proposed models were inspired by the original Deep Complex model and by models participating in MIREX, where the latter provided inspiration to add a recurrent layer to the network, and to use the CQT. It is interesting how the LSTM layer by itself (by keeping the FFT representation) worsened the model while adding the CQT together with the LSTM layer improved it. One reason could be that the LSTM layer increased the number of parameters and the amount of 100 epochs in the training section was not enough for the FFT model in order to find the optimal weights. 100 epochs seemed to be enough for the CQT model which can be explained by the CQT consisted of 1/4 the input data as the FFT model used. The reason why the CQT did improve over the Deep Complex model is probably because of the convolutional layer, in which it is advantageous to convolve the same number of frequency bins between each musical note. A convolution of an FFT representation with linearly spaced frequency bins might convolve frequencies corresponding to two different musical notes, which would then not be possible to distinguish.

The Deep Complex LSTMCQT model achieved the 3rd best accuracy score in the Multi-F0 task. Despite that, a couple of possible improvements are possible. First of all, the model was trained to optimize the accuracy, (44), where the true negatives were counted in the score. Instead, the MIREX accuracy in (43) should be used. The second suggested improvement is about postprocessing, where a more advanced method than the binary threshold should probably be used. From the plots of the estimates it was clear that the model often estimated the second harmonic as a separate fundamental frequency. Using a more specialized algorithm could probably reduce the amount of false positives. For future work it would also be of interest to change the architecture of the neural network and change some of the convolutional layers[79]. Some feature detectors could be added to the model, such as, for example, an onset detector and a beat detector. Finally, the training data may be randomly shifted in the frequency domain, in order to reduce overfitting.

8.4 Ideas for Future Research

A couple of ideas emerged while typing this thesis. One was about the recurrent layers in a neural network which are dependent of the previous data. Unlike other application areas of the recurrent layers, the input data for the next time bins is known. Therefore the future input data could be used for predicting the frequencies in the previous time step, as a reversed recurrent layer. A version of this exists in convolutional neural network where spectrums from previous and future time bins are convolved, but the idea is that there might be even longer dependencies in the future as well as the past. A combination of a neural network and a Hidden Markov model might be used for this purpose.

Another idea was about a big issue in the music transcription research area, namely the lack of available datasets consisting of instrumental music and its annotated sheet music. This thesis shows that some models, mainly parametric and probabilistic, are performing differently on real recorded music and simulated music. For those models, which generally are older, the type of data is crucial both for training and testing. This thesis also shows that other models, mainly neural network models, perform equally on real and simulated music. If a study would consider the same test files in two versions, both real and simulated music, in order to investigate if one

[79] As the THK1 performed very well on fewer amount of hidden layers.

can prove that the neural network based methods are performing equally on the different types of datasets, thousands of datasets with perfectly annotated sheet music will be available very cheaply. Also, the sheet music annotations would be perfectly aligned with the sound files, which is currently not possible to achieve manually. This would break a lot of limitations which the research field has been struggling with for a long time.

9 Bibliography

References

[1] I. Aizenberg and A. Gonzalez, "Image Recognition using MLMVN and Frequency Domain Features," *2018 IEEE International Joint Conference on Neural Networks (IJCNN)*, July, 2018 pp. 1550-1557. USA, July 2018. [Online]. Available: `https://drive.google.com/file/d/1CR6ijceulhdPqrQa5yrii6V9MIU3FKlM/view`. [Accessed August 16, 2018].

[2] K. Dressler, "Automatic Transcription of the Melody from Polyphonic Music," Ph.D. dissertation, Fak. für Elekt. und Infor. der Techn. Univ. Ilmenau, 2016. [Online]. Available: `https://www.db-thueringen.de/servlets/MCRFileNodeServlet/dbt_derivate_00038847/ilm1-2017000136.pdf`. [Accessed August 16, 2018].

[3] E. Benetos, S. Dixon, D. Giannoulis et al. "Automatic music transcription: challenges and future directions," *J Intell Inf Syst* 41: 407, 2013. [Online]. Available: `https://doi.org/10.1007/s10844-013-0258-3`. [Accessed August 16, 2018].

[4] "Sound," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Sound`. [Accessed August 16, 2018].

[5] "Fast Fourier transform," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Fast_Fourier_transform`. [Accessed August 16, 2018].

[6] J. Florén and J. Torby, "Musical Instrument Categorization using Spectral- and Cepstral Analysis," B.S. thesis, Center of Math. Scien. Lund Univ., 2016. [Online]. Available: `http://lup.lub.lu.se/student-papers/record/8891409`. [Accessed August 16, 2018].

[7] "Complex number," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Complex_number`. [Accessed September 1, 2018].

[8] F. Galletta, "Do the 88 keys of a piano cover the full range of frequencies that the human ear can perceive?," 2017. [Online forum image] Available: `https://www.quora.com/Do-the-88-keys-of-a-piano-cover-the-full-range-of-frequencies-that-the-human-ear-can-perceive`. [Accessed August 16, 2018].

[9] "Musical note," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Musical_note`. [Accessed September 1, 2018].

[10] J.C. Brown, "Calculation of a Constant Q spectral transform," *Journal of the Acoustical Society of America*, vol. 144, no. 1, pp. 425-434, Jan. 1991. [Online]. Available: `https://doi.org/10.1121/1.400476`. [Accessed August 16, 2018].

[11] B.C.J. Moore and B.R. Glasberg, "Suggested formulae for calculating auditory-filter bandwidths and excitation patterns," *Journal of the Acoustical Society of America*, vol. 74, no. 3, pp. 750-753, 1983.

[12] D. O'Shaughnessy, "Speech communication: human and machine", *Addison-Wesley*, 1987, pp. 150. [Online]. Available: `https://books.google.se/books?ei=rhFfSpa-BJOCNsTT4IUG&id=mHFQAAAAMAAJ&dq=Speech+Communications:+Human+and+Machine&q=2595&redir_esc=y#search_anchor`. [Accessed August 16, 2018].

[13] C.M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag Berlin, Heidelberg, 2006.

[14] G. Sanderson. "Neural Networks," 2017-2018 [Online video playlist]. Available: `https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi`. [Accessed August 16, 2018].

[15] L-C. Böiers. *Lectures on Optimization*, 3rd ed. KFS AB. 2009.

[16] "Gradient descent," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Gradient_descent`. [Accessed October 4, 2018].

[17] A. Karpathy, "Convolutional Neural Networks for Visual Recognition," *Lecture at Stanford Univ*, spring 2018. [Online]. Available: `http://cs231n.github.io/convolutional-networks/` [Accessed August 16, 2018].

[18] "MIREX HOME," *Mirex*. [Online]. Available: `http://www.music-ir.org/mirex/wiki/MIREX_HOME`. [Accessed August 16, 2018].

[19] C. Olah, "Understanding LSTM Networks," [Online]. Available: `http://colah.github.io/posts/2015-08-Understanding-LSTMs/`. [Accessed August 16, 2018].

[20] C. Trabelsi, O. Bilaniuk, D. Serdyuk, S. Subramian and J.F. Santos et al. "Deep Complex Networks," *ICLR*, 2018.[Online]. Available: `https://arxiv.org/abs/1705.09792`. [Accessed August 16, 2018].

[21] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift," Google Inc. 2015. [Online]. Available: `https://arxiv.org/abs/1502.03167`. [Accessed September 9, 2018].

[22] V. Emiya, MAPS Database – A piano database for multipitch estimation and automatic transcription of music, 2008. [Online]. Available by request: `http://www.tsi.telecom-paristech.fr/aao/en/2010/07/08/maps-database-a-piano-database-for-multipitch-estimation-and-automatic-transcription-of-music/`. [Accessed August 16, 2018].

[23] J. Thickstun, Z. Harchaoui and S. Kakade, MusicNet, 2016. [Online]. Available: `https://homes.cs.washington.edu/~thickstn/musicnet.html`. [Accessed August 16, 2018].

[24] Z. Duan, B. Pardo and C. Zhang, Bach10 Dataset, "Multiple fundamental frequency estimation by modeling spectral peaks and non-peak regions," *IEEE Trans. Audio Speech Lang. Process.* vol. 18, no. 8, pp. 2121-2133, 2010. [Online]. Available by request: `http://music.cs.northwestern.edu/data/Bach10.html`. [Accessed August 16, 2018].

[25] RWC Music Database, 2002. [Online]. Available by request: `https://staff.aist.go.jp/m.goto/RWC-MDB/`. [Accessed August 16, 2018].

[26] MIREX Development Dataset, 2005. Available by request: `http://www.music-ir.org/mirex/wiki/2017:Multiple_Fundamental_Frequency_Estimation_%26_Tracking`. [Accessed August 16, 2018].

[27] L. Su and Y. H. Yang, "Escaping from the Abyss of Manual Annotation: New Methodology of Building Polyphonic Datasets for Automatic Music Transcription," *Int. Symp. Computer Music Multidisciplinary Research (CMMR)*, June 2015. [Online]. Available: `https://sites.google.com/site/lisupage/research/new-methodology-of-building-polyphonic-datasets-for-amt`. [Accessed August 16, 2018].

[28] C. Focacci, "False Negatives: A Serious Danger in Your AML Program", April, 2016. [Online image]. Available: `https://transparint.com/blog/2016/04/01/false-negatives-a-serious-danger-in-your-aml-program/`. [Accessed August 16, 2018].

[29] Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang and D. P. W. Ellis, "mir_eval: A Transparent Implementation of Common MIR Metrics", *Proceedings of the 15th International Conference on Music Information Retrieval*, 2014. [Online]. Available: `http://colinraffel.com/publications/ismir2014mir_eval.pdf`. [Accessed August 16, 2018].

[30] "Precision and recall," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Precision_and_recall`. [Accessed August 16, 2018].

[31] A. Berg and S. Blomstrand. "Pitch Perfect: Building a MIREX Evaluation System," Project in Stationary and Non-Stationary Spectral Analysis, Lund Univ.

[32] "Tukey's range test," *Wikipedia*. [Online]. Available: `https://en.wikipedia.org/wiki/Tukey%27s_range_test`. [Accessed August 16, 2018].

[33] E. Benetos and T. Weyde, "Multiple-F0 Estimation and Note Tracking for MIREX 2015 Using a Sound State-Based Spectrogram Factorization Model," *MIREX 2015*. [Online]. Available: `http://www.music-ir.org/mirex/abstracts/2015/BW1.pdf`. [Accessed August 16, 2018].

[34] C. Cannam, E. Benetos, M. Mauch, M. E. P. Davies, S. Dixon, C. Landone, K. Noland and Dan Stowell, "MIREX 2015: Vamp Plugins from the Centre for Digital Music," *MIREX 2015*. [Online]. Available: `http://www.music-ir.org/mirex/abstracts/2015/CB1.pdf`. [Accessed August 16, 2018].

[35] E. Benetos and S. Dixon, "A Shift-Invariant Latent Variable Model for Automatic Music Transcription," *Computer Music Journal*, 36(4), pp. 81-94. [Online]. Available: `http://dx.doi.org/10.1162/COMJ_a_00146`. [Accessed August 16, 2018].

[36] L. Su and Y. H. Yang, "Multiple-F0 Estimation for MIREX 2015," *MIREX 2015*. [Online]. Available: `http://www.music-ir.org/mirex/abstracts/2015/SY1.pdf`. [Accessed August 16, 2018].

[37] G. Peeters, "Music pitch representation by periodicity measures based on combined temporal and spectral representations," *2006 IEEE Inter. Conf. on Acoustics Speech and Signal Processing Proceedings*, 2006. [Online]. Availible: `http://recherche.ircam.fr/anasyn/peeters/ARTICLES/Peeters_2006_ICASSP_Periodicity.pdf`. [Accessed August 16, 2018].

[38] D. Troxel, "Music Transcription with a Convolutional Neural Network 2016," *MIREX 2016*. [Online]. Available: `http://www.music-ir.org/mirex/abstracts/2016/DT1.pdf`. [Accessed August 16, 2018].

[39] K. He, X. Zhang, S.Ren, J. Sun, "Deep Residual Learning for Image Recognition," Microsoft Research, 2015. [Online]. Available: https://arxiv.org/abs/1512.03385. [Accessed August 16, 2018].

[40] M. Marolt, "A Connectionist Approach to Automatic Transcription of Polyphonic Piano Music," Fac. of Comp. and Info. Sci. at Univ. of Ljubljana, 2004. [Online]. Available: http://lgm.fri.uni-lj.si/wp-content/uploads/2016/09/4203860.pdf. [Accessed August 16, 2018].

[41] C. Thomé and S. Ahlbäck "Polyphonic Pitch Detection with Convolutional Recurrent Neural Networks," *MIREX 2017*. [Online]. Available: http://www.music-ir.org/mirex/abstracts/2017/CT1.pdf. [Accessed August 16, 2018].

[42] S. Mita, G. Hatanaka, A. Meneses, N. Thammasan and D. Miura, "MIREX 2017 : Multi-instrumental End-to-End Convolutional Neural Network for Multiple F0 estimation," *MIREX 2017*. [Online]. Available: http://www.music-ir.org/mirex/abstracts/2017/MHMTM1.pdf. [Accessed August 16, 2018].

[43] L. Pogorelyuk and C. Rowley, "Melody Extraction for MIREX 2016 Using Dynamic Mode Decomposition," *MIREX 2017*. [Online]. Available: http://www.music-ir.org/mirex/abstracts/2017/PR1.pdf. [Accessed August 16, 2018].

[44] J. Thickstun, Z. Harchaoui, D. Foster and S. M. Kakade, "MIREX 2017: Frequency Domain Convolutions for Multiple F0 Estimation," *MIREX 2017*. [Online]. Available: http://www.music-ir.org/mirex/abstracts/2017/THK1.pdf. [Accessed August 16, 2018].

[45] J. Thickstun, Z. Harchaoui and S. M. Kakade, "Learning Features of Music From Scratch," *ICLR 2017*. [Online]. Available: https://arxiv.org/pdf/1611.09827.pdf. [Accessed August 16, 2018].

[46] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore and D. Eck, "Onsets and Frames: Dual-Objective Piano Transcription," Google Brain Team, Mountain View, CA, USA, 2017. [Online]. Available: https://arxiv.org/abs/1710.11153. [Accessed August 16, 2018].

[47] T. Kronvall, F. Elvander, S. Ingi Adalbjörnsson and Andreas Jakobsson, "An Adaptive Penalty Approach to Multi-Pitch Estimation," *EUSIPCO 2015*. [Online]. Available: https://www.eurasip.org/Proceedings/Eusipco/Eusipco2015/papers/1570103397.pdf. [Accessed August 16, 2018].

[48] T. Tolonen and M. Karjalainen, "A Computationally Efficient MultipitchnAnalysis Model," *IEEE Transactions on Speech and Audio Processing*, vol. 8, Issue 6, pp. 708 - 716, Nov 2000. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.334.1573&rep=rep1&type=pdf. [Accessed August 16, 2018].

[49] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," *ICLR*, April 2015. [Online]. Available: https://arxiv.org/abs/1409.1556. [Accessed August 16, 2018].

[50] J. Thickstun, Z, Harchaoui, D. Foster, S. M. Kakade,"Invariances and Data Augmentation for Supervised Music Transcription," 2017. [Online]. Available: https://arxiv.org/abs/1711.04845. [Accessed August 16, 2018].

[51] MuseScore. [Online]. Available: https://musescore.org/en. [Accessed August 16, 2018].

[52] readmidi, Matlab function. [Online]. Available: http://kenschutte.com/midi. [Accessed August 16, 2018].