Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Henrik Grønbech

Multi-Instrument Automatic Music Transcription with Deep Learning

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

Henrik Grønbech

Multi-Instrument Automatic Music Transcription with Deep Learning

Master's thesis in Computer Science
Supervisor: Björn Gambäck
June 2021

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

NTNU
Norwegian University of
Science and Technology

Abstract

Automatic music transcription (AMT) is the task of using computers to turn audio of music into a symbolic representation such as Musical Instrument Digital Interface (MIDI) or sheet music. This task can be regarded as the musical analog of speech-to-text and the symbolic representation is at least as useful as written text is for natural language. Sheet music enables musicians to learn new pieces and can aid during live performances. Digital music representations can be used to remix music, create new arrangements, and analyze chord progressions and other musical structures. Automatic music transcription has long been regarded as one of the most difficult tasks in signal processing, but with the progress in deep learning the performance in a single-instrument setting on piano is almost solved with a state-of-the-art note F_1 score of 96.72.

The goal of this Master's Thesis is to extend this to a multi-instrument setting and several experiments have been conducted. The first set of experiments investigates different architectures and music source separation pre-processing for multi-instrument AMT. These experiments show that the current single-instrument AMT model works well on a multi-instrument audio source, and can be further enhanced by using a joint music source separation and automatic music transcription architecture. Music source separation pre-processing did not improve performance, but the model was not fine-tuned on the used dataset.

Another experiment shows that it is possible to train a universal note-level AMT model solely on a mixture audio source. This model reaches a note F_1 scores of 90.6 on piano and 95.8 on bass audio, only slightly behind the current state-of-the-art in the single-instrument setting. The transcription performance varies greatly between instrument classes and the note-with-offset scores are still far behind the current single-instrument for all instrument classes except bass.

Finally, a stream-level model is trained that is able to transcribe piano, guitar, bass, drum and all the pitched instruments simultaneously in 5-10 times real-time performance on CPU and 10-100 times real-time performance on GPU.

All the experiments are conducted on the synthetically rendered MIDI dataset Slakh. During the work on this dataset, several systematic and non-systematic errors were found and reported to the creators of the dataset. An efficient and convenient PyTorch data-loader is created for this dataset which addresses these errors and enables reproducibility.

Sammendrag

Automatisk transkribering av musikk går ut på å bruke datamaskiner til å transformere lydfiler til en symbolsk representasjon, som MIDI-filer («Musical Instrument Digital Interface») eller noter. Denne oppgaven er den musikalske versjonen av tale til tekst og er vel så nyttig som tekst er for naturlig språk. Noter hjelper musikere å lære musikk og brukes også under fremføringer. Digitale representasjoner av musikk kan brukes til å remikse musikk eller lage nye arrangementer og for å analysere akkordprogresjoner og andre strukturer i musikken. Automatisk transkribering av musikk har lenge blitt sett på som en av de vanskeligste oppgavene innenfor digital signalbehandling, men med utviklingen av dyp læring har problemet nesten blitt løst for piano med en state-of-the-art note-F_1-verdi på 96,72.

Målet med denne masteroppgaven er å utvide transkriberingen til et flerinstrument-miljø. Den første gruppen eksperimenter i denne oppgaven undersøker ulike arkitekturer og effekten av å separere lydfilene med eksisterende modeller på forhånd. Disse eksperimentene viser at den eksisterende enkeltinstrumentarkitekturen fungerer godt i et flerinstrumentmiljø. Resultatene blir enda bedre med en kombinert musikkseparerings- og transkriberingsarkitektur. Separering av lydfilene på forhånd ga ikke bedre resultater, men modellen var heller ikke finjustert på datasettet brukt i eksperimentene.

Et annet eksperiment viser at det er mulig å trene en universell transkriberings-modell. Denne modellen er trent på lydfiler av et fullt band og klarer å transkribere enkeltinstrumenter med en note-F_1-verdi på 90,6 på piano og 95,8 bass – rett bak state-of-the-art-verdiene for piano. Resultatene varierer likevel mye mellom ulike instrumenttyper, og note-med-slutt-resultatene ligger langt bak state-of-the-art for alle instrumenttypene utenom bass.

I det siste eksperimentet er det trent en modell som transkriberer alle instrumenter på én gang og klassifiserer notene som piano, gitar, bass, trommer og annet. Denne modellen kjører i 5-10 ganger sanntid på CPU og 10-100 ganger på GPU.

Alle eksperimentene er utført på det MIDI-genererte datasettet Slakh. Gjennom arbeidet med dette datasettet har flere feil ble funnet og rapportert til de som lagde datasettet. En effektiv datalaster i maskinlæringsverktøyet PyTorch har blitt laget som tar høyde for disse feilene og gjør det lett for andre å reprodusere eksperimentene.

Preface

This Thesis is submitted as the final part to achieve a Master's degree in Computer Science with specialization in *Artificial Intelligence* from the Norwegian University of Science and Technology (NTNU). The work was done at the Department of Computer Science and was supervised by Björn Gambäck. The work equates to 30 ECTS-credits, equal to one semester.

The Thesis is based on the work from my specialization project, *Automatic music transcription with deep learning* (Grønbech, 2020), and chapters, sections, formulations, and figures are taken or adapted from that report. These sections will be indicated at the beginning of each chapter.

Special thanks go to Björn for encouraging me to follow my passion for music and computer science/artificial intelligence and letting me choose this research project. I am grateful for the discussions and thorough feedback on the Thesis.

I would like to thank Emmanouil Benetos for permitting me to reprint Figure 2.1 and to Wikimedia Commons, Hawthorne et al. (2018), Kim and Bello (2019), Stoller et al. (2018) and Jansson et al. (2017) for releasing their work under a CC BY-SA 4.0 license[1] which enables me to reprint their figures.

The work would not be possible if Raffel (2016) had not laid the groundwork by creating the Lakh dataset and Manilow et al. (2019) the Slakh dataset. I appreciate that Ethan Manilow has read and answered my GitHub issues and plans to release a new version of the Slakh dataset based on the errors found in this work.

I appreciate the high-quality work by Kim and Bello (2019) and releasing the Onset and Frames source code[2] under the MIT license[3]. The code bases developed in this Thesis are based on that work.

Furthermore, a thanks goes out to the HPC group at NTNU, for allowing the use of the Idun cluster (Själander et al., 2019). The experiments in this Thesis would not have been possible without these resources.

In addition, I would like to thank everyone that have done work in the field of automatic music transcription and music source separation. Nothing would give me greater pleasure if the work in this Thesis enables the field to move further.

Finally, I would like to express thankfulness to Mathias Bynke for helping with the Norwegian translation of the abstract, and family, friends and loved ones for supporting my work.

Henrik Grønbech
Trondheim, 11th June 2021

[1] https://creativecommons.org/licenses/by-sa/4.0
[2] https://github.com/jongwook/onsets-and-frames
[3] https://opensource.org/licenses/MIT

Contents

List of Figures

List of Tables

1. Introduction

Automatic music transcription (AMT) is the task of using computers to turn audio of music into a symbolic representation such as Musical Instrument Digital Interface (MIDI) or sheet music. In other words, given an audio file, we want the computer to extract the pitches and note durations from a musical piece. For stream-level transcriptions also the instruments that play each note are extracted.

There was little to no progress in this field for a long time, but with the rise of deep learning and new datasets, improvements have accelerated. For a single polyphonic instrument, the piano, this task is now almost solved with a recent onset F1-score of 96.72. This Thesis extends this to a multi-instrument setting. The synthetically rendered MIDI dataset Slakh is used in all the experiments.

1.1. Background and Motivation

Transcribing music is the process of listening to a piece of music audio, extracting the notes the music consists of and writing down in a symbolic form such as sheet music or into a music notation software. To an untrained ear, this can be challenging even if the audio only consists of a single instrument that plays one note at a time in a clearly audible frequency range. For polyphonic instruments, such as a piano, the task quickly becomes almost impossible unless you have perfect pitch or several years of experience—and even for professional musicians, the task is error-prone and very time-consuming. A typical band song consists of drums, bass, guitar, piano, and vocals. Some instruments might have several tracks and some tracks or instruments can be mixed very low in the audio. A fully accurate transcription would not only capture all the instruments in the song but what each instrument plays and how loud it plays each note. In other genres, such as classical symphonies, there might be several dozens of instruments at a time. Given the immense difficulty of this problem for humans, there is no wonder why the automatic version of this task has been known to be one of the most difficult tasks in the field of *music information retrieval* (MIR) and *signal processing.*

Applications of music in a symbolic form are vast. Many musicians, especially those that are classically trained, are used to learn musical pieces only from sheet music. If they want to play in another setting, such as in a rock band where the musicians are more used to learn music by ear, an automatic music transcription system would be highly valuable. This would also open up a lot of possibilities for musicians wanting to learn songs that do not have any sheet music yet. Likewise, arranging music for an ensemble based on existing music audio would also benefit from automatic music transcription software. Not only is it very time-consuming, but can be very difficult to get started without the necessary experience. A high-quality automatic music transcription system would democratize this to more people and drastically reduce arranging time.

Other applications of automatic music transcriptions software include those of automatically creating transcriptions for games such as GuitarHero, RockBand or SingStar. It can also be used when remixing songs and share musical ideas, just like it is easier to send messages with text to each other than voice recordings. Furthermore, automatic music transcription also has applications in live performances. It would open up possibilities to mix acoustic and electric sounds. One could for instance augment the sound from an acoustic piano with digital sound from a synthesizer in real-time.

An automatic music transcription model can reach super-human performance transcribing music at many times real-time performance. Humans are inherently limited by the speed we can perceive sounds—computer does not have this limitation. An automatic music transcription model could be used as an pre-proseccing step to analyze music at a large scale. Chords progressions, common bass-lines, and other musical structures can be extracted from the audio. AMT systems could also be used to pattern match audio which again can, for instance, be used for musical copyright infringement systems.

1.2. Goals and Research Questions

This Master's Thesis has one goal that will be described below. To reach this goal, five Research Questions have been composed.

Goal 1 *Introduce multi-instrument automatic music transcription models.*

Transcribing piano and drums on a single-instrument audio source has reached high performance in the last couple of years. This Master's Thesis will try to expand this to a multi-instrument setting. Multi-instrument in this context has two meanings, the first is a model that can transcribe several instruments simultaneously, and the other is a model that can transcribe a single instrument in a multi-instrument audio source. Both of these cases will be investigated in the Research Questions. As music source separation also has gained a lot of momentum recently it will be investigated if it can be a useful pre-processing step for automatic music transcription. The goal of this thesis will be reached by answering the following research questions.

Research Question 1–3 will operate in the first multi-instrument case, namely, transcribing a single instrument in a multi-instrument audio source.

Research question 1 *How well do state-of-the-art single-instrument models perform in a multi-instrument setting without music source separation?*

To answer this Research Question a benchmark on a single-instrument audio source will be created for comparison. The training process in this step will be very similar to a single-instrument setting, however, the audio source will be changed with an audio source containing several instruments.

Research question 2 *How well do state-of-the-art single-instrument models perform in a multi-instrument setting when the audio is source separated by a pre-trained music source separation model?*

To answer this Research Question, a music source separation model will separate the instrument of interest before the model is trained an evaluated. Music source separation models typically separate music into vocals, drums, bass and accompaniment. When the audio is separated, it should be easier for the automatic transcription models to focus on the instrument that will be transcribed (this is equal to the single-instrument setting). However, music source separation models still have some deficiencies so the transcription model will need to take into account.

Research question 3 *How well does a new architecture that joins source separation with automatic music transcription perform in a multi-instrument setting?*

This research question is connected to Research Question 2. Two considerations have to be taken into account; will the new architecture be trained jointly on both source separation *and* automatic music transcription or will the model only be trained on automatic music transcription, but have a similar architecture that resembles the source separation models.

Research question 4 *How well does a note-level multi-instrument automatic music transcription perform in a single-instrument setting?*

A note-level multi-instrument model is a model that is able to transcribe several instruments, but not able to tell which instrument played a given note. After the model is trained, it will be evaluated on the same benchmark as the baseline described under Research Question 1 as well as several other instrument classes.

Research question 5 *How well does a stream-level multi-instrument automatic music transcription perform?*

A stream-level multi-instrument model is a model that is not only able to transcribe several instruments but also tell which instrument played a given note. To answer this Research Question a stream-level multi-instrument has to be engineered and trained. It will be evaluated on the results from Research Question 1–3.

1.3. Research Method

To achieve the goal of the Master's Thesis, an experimental research methodology will be applied. The experiments will be trained and evaluated on the synthetically rendered MIDI [1] dataset Slakh (Manilow et al., 2019). The common metric in the field of automatic music transcription, provided by the Python library `mir_eval` (Raffel et al., 2014), will

[1]Musical Instrument Digital Interface

be used to evaluate the models. Since several experiments will be performed on the same dataset, a transparent comparison between the models should be possible. Regarding Research Questions 3, the construction of the models will be subject to a large degree of experimentation and mostly quantitatively evaluated. However, other aspects of the models such as training time, inference time and memory size will influence the design process.

1.4. Contributions

The main contributions of this Master's Thesis are as follows:

1. *An investigation of different architectures for multi-instrument automatic music transcription. This work shows that it is possible to use a state-of-the-art single-instrument model on a multi-instrument source with promising results. The results are further improved by using the popular music source separation model U-Net as a backbone.*

2. *Experiments on using music source separation as a pre-processing step for automatic music transcription.*

3. *A universal note-level automatic music transcription model is trained that can transcribe individual instrument classes with promising results. An investigation of which instrument classes are most easily transcribed are also shown.*

4. *A stream-level automatic music transcription model that can transcribe piano, guitar, bass, drum and all pitched instrument simultaneously is trained. This model efficiently transcribes all the instruments commonly seen in a pop or rock band in around 5-10 real-time performance on CPU and 10-100 times real-time performance on GPU.*

5. *A PyTorch data-loader that efficiently and conveniently loads audio and labels from the Slakh dataset is created. Several systematic and non-systematic errors in the dataset are accounted for in this work and reported to the creators of the dataset.*

1.5. Thesis Structure

The Master's Thesis is structured in the following manner:

1. Chapter 2 provides the background theory necessary to understand the rest of the Thesis. The chapter includes topics such as music information retrieval, representations of audio and Fourier transformations, a brief introduction to machine learning and deep learning, and the evaluation metrics used for automatic music transcription.

2. Chapter 3 presents the dataset used for automatic music transcription, music source separation and music information retrieval. Eight datasets are presented, where five datasets are suitable for automatic music transcription.

3. Chapter 4 covers the related work carried out in the field of AMT and music source separation. First, an overview of traditional automatic music transcription models is given before the state-of-the-art deep-learning-based approaches.

4. Chapter 5 covers the model architectures which will be used for the different experiments.

5. Chapter 6 covers an experimental plan, the experimental setup and the results of the experiments in this work.

6. Finally, chapter 7 evaluates and discusses the Master's Thesis in light of the results and findings of the experiments. Contributions and suggestions for possible further work are presented at the end.

2. Background Theory

This chapter covers the theory and background behind the different areas relevant to this project. First, a general introduction to the field of music information retrieval with an emphasis on automatic music transcription and music source separation will be given. The next section gives an overview of audio and different representations of music. The following section gives a brief introduction to machine learning as well as relevant architectures. The final section consists of theory and approaches for evaluation.

The following sections is based on the work in Grønbech (2020); 2.1.1 is from Section 2.1 with minor modifications; section 2.2 is from 2.2, 2.3 is from 2.3 and, 2.4 is from 2.4 and 2.5 is from Section 2.5 without any modifications.

2.1. Music Information Retrieval

Music information retrieval (MIR) is the field of study that is concerned with retrieving information from music. There exist many different subdisciplines of MIR such as music classifications, recommender systems, music source separation, instrument recognition, automatic music transcription and automatic music generation. Music classification consists of genre classification (categorizing music into genres such as rock, jazz classical, etc.), artist classification and mood classification. A music recommender system is a system that tries to predict the rating a listener would give to a given piece of music. This can be useful for streaming services such as Spotify[1] to recommend music, for record labels or even for musicians to make music more people like. An overview of the field of automatic music transcription will be given in section 2.1.1 and music source separation and instrument recognition will be given in section 2.1.2.

2.1.1. Automatic Music Transcription

Automatic music transcription is the task of converting audio of music into some form of music notation. Depending on the application, the notation can be sheet music (an example is shown in Figure 2.1(d)), guitar tablature, or Musical Instrument Digital Interface (MIDI) files (see Section 2.2.1). Some of these notation requires a higher level of understanding of the music than the other. Sheet music, for instance, does not only require the pitches but also the time signature, number of voices, and key signature. Automatic music transcription is regarded as one of the most difficult tasks in field of signal processing therefore this task has been divided into subtasks of different degree

[1]https://www.spotify.com/

of difficulty. Categorized by Benetos et al. (2019), the four levels of transcription are frame-level, note-level, stream-level, and score-level.

Frame-level transcription, also called *Multi-Pitch Estimation (MPE)*, is the task of estimating the fundamental frequencies in a given time frame. The length of the frame is usually on the order of 10ms. The frames can be estimated independently of each other.

Note-level transcription, or *note tracking*, is one level of abstraction higher than frame-level transcription. In this level of transcription, we are interested in the note onsets (when a note begins) and the duration of the note. Each frame can no longer be classified independently. A longer context, at least in the order of seconds, is needed.

Stream-level transcription, also called *Multi-Pitch Streaming (MPS)*, is yet one higher level of transcription. Groups of notes are estimated into a stream, which typically corresponds to one instrument or a voice in a choir. When transcribing into this level a model can no longer just find the fundamental frequencies of each note, the timbre and overtones must also be taken into consideration.

Score-level Transcription is the highest level of transcription. This level aims to transcribe into a human-readable musical score. Transcription at this level requires a deeper understanding of musical structures. Rhythmic structures such as beats and bars help to quantize the lengths of notes. Stream structures aid the assignment of notes to different musical staffs.

2.1.2. Music source separation

Music source separation is the task of separating the individual tracks or instruments that makes up a piece of music. It is useful to separate the music into instruments such as bass, drums, piano, vocals and the rest of the music. This task is both useful for remixing music, but can also be used in an education setting for learning what each instrument plays. Likewise, music source separation can be used as a pre-processing step for automatic music transcription.

2.2. Audio and representations of music

Sound is variations in pressure in a medium. The most common medium we humans perceive sound in is air but it can also be others such as water or through our bones. The frequency in these pressure variations determines the tone and the magnitude difference in the high and low pressure-periods determines the volume. We humans experiences differences in frequencies logarithmic—the note A on the middle of the piano has a frequency of 440Hz while the A an octave above has a frequency of 880Hz and the next 1,760Hz. Our ears can pick up frequencies between approximately 20Hz to 20,000Hz (this, unfortunately, decays somewhat with age), but the fundamental frequency on the piano is only between 27.5Hz to 4,186Hz. The fundamental frequency determines the pitch while the overtones change the timbre. Overtones, also called partials, are usually at integer multiples of the fundamental frequencies. In Figure 2.1(b) at the first note after three seconds, we see that the fundamental frequency around 330Hz is strongest

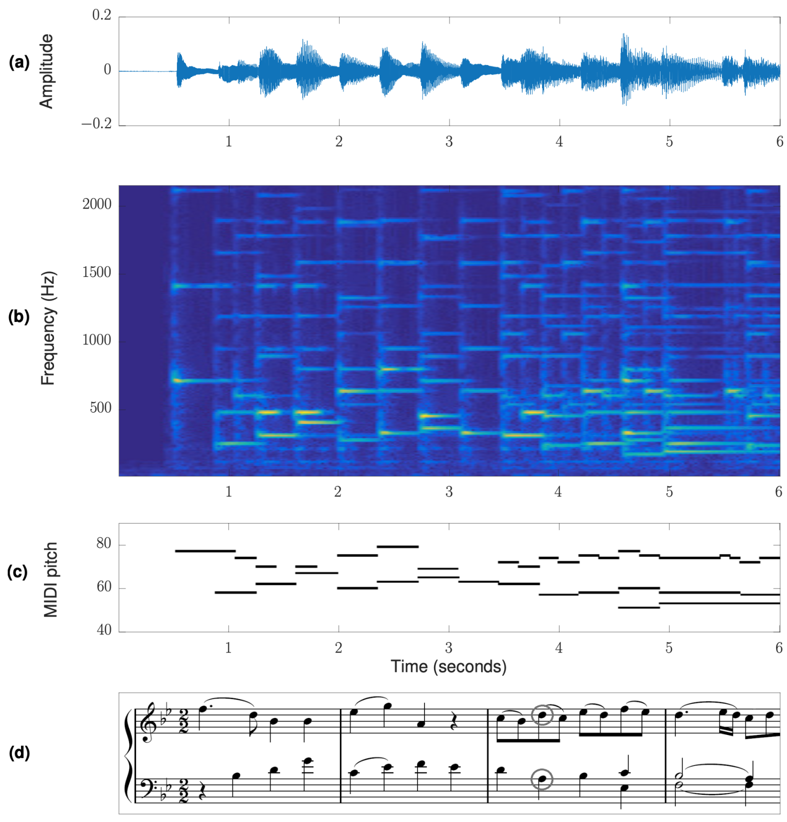

Figure 2.1.: Different representations of music. (a) Waveform in time domain, (b) Time-frequency representation, (c) Piano-roll representation, (d) Sheet music. The example corresponds to the first 6 seconds of W. A. Mozart's Piano Sonata No. 13, 3rd movement (taken from the MAPS database). Reprint of Figure 1 from Benetos et al. (2019) with permission to reuse.

while the overtones are linearly spaced above.

Audio is digitally stored as a time series where each value represents a pressure difference. This representation is often called the *raw audio domain* or *waveform domain*. The *sample rate* is how many times the value is measured per second and *bit depth* is the number of bits of information in each sample. A CD quality recording typically has a sample rate of 44,100Hz and a bit depth of 16.

Four different representations of music are shown in Figure 2.1. At the top, we see a waveform domain representation as it would be stored digitally. Next, a time-frequency representation is shown (see section 2.2.2 for more information). Thirdly, a matrix-like piano roll representation is shown. This is typically how an automatic music transcription system would output the data and this representation is often used in Digital Audio Workstations. Lastly, sheet music that is normally used by musicians for performing music is shown.

2.2.1. MIDI

Musical Instrument Digital Interface (MIDI) is an open format for communicating and storing musical signals. It was introduced in the 1980s to synchronize synthesizers which previously had used proprietary communication formats. The format includes electrical connections, messages, and a file format. The messages are events based—pitches has an on and an off message, and there are messages for velocity, vibrato/pitch bend, panning, and clock signals.

One limitation of the format is that the pitch bend message is global for all the pitches. This makes it impossible to create micro-tonal music. A new standard, the MIDI 2.0 standard, was announced at the 2020 Winter NAMM show [2] which among other improvements includes this ability.

2.2.2. Fourier transformation

It is possible to transform audio in the time domain to a frequency domain by a *Fourier transformation*. The idea in this transformation is that we want to find the presence of different frequencies in the raw time-domain signal. This can be achieved by calculating the "center of mass" in the complex plane by multiplying the time domain signal with e^{if}, where f denotes the frequency, and summing/integrating the values. A beautiful animation and explanation can be seen in this video by 3Blue1Brown [3].

Mathematically, this can be expressed in the discrete case by

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{i2\pi}{N}kn}, \tag{2.1}$$

where $x = \{x_0, x_1, ..., x_{N-1}\}$ is the input signal and $X = \{X_0, X_1, ..., X_{N-1}\}$ is the transformed frequencies.

This transformation, however, loses all the temporal information in the original signal. Due to this, the *short-time discrete Fourier transform* is often applied to music. In this transformation, the input signal $x = \{x_0, x_1, ..., x_{N-1}\}$ is divided into smaller chunks. The size of the chunk is called *window size* and the number of values between the start of each chunk is called *step size*. Due to artifacts, the step size is usually smaller than the window size. Each of these chunks is multiplied by a *window function*, such as a Hann window or Gaussian window, and then discrete Fourier transformation as in equation 2.1 is calculated on each chunk.

2.2.3. Mel-scaled Spectrogram and Constant-Q Transform

The *short-time discrete Fourier transform* described in the last section creates a *time-frequency representation* where the frequencies are linearly spaced. Since we humans perceive sound logarithmically, the frequencies can be scaled to reflect this. This is called

[2]https://www.midi.org/midi-articles/midi-2-0-at-the-2020-namm-show
[3]https://www.youtube.com/watch?v=spUNpyF58BY

a *mel-scaled spectrogram*. This representation will have a lower frequency resolution in the lower frequencies.

There is another time-frequency transformation that keeps the same frequency resolution in the logarithmic scale called the *Constant-Q transform*. The window length is longer for lower frequencies and shorter for higher frequencies in the transformation. Otherwise, it very similar to the short-time discrete Fourier transform.

2.3. Machine Learning

Machine Learning is a subfield of Artificial Intelligence. It is the study and exploration of algorithms that can learn from data. In other fields, a programmer would need to tell the computer how to solve a given task, while in machine learning the computer learns to solve the task from data and previous examples. As such, machine learning enables us to solve problems that have been hard or impossible for humans to handcraft rules.

Machine learning algorithms are generally divided into three categories—supervised, semi-supervised and unsupervised learning. In supervised learning, the models have labels for the problem at hand in the training process. In automatic music transcription, the label would typically be MIDI files with the pitches and durations for each note. Semi-supervised learning combines a small amount of labeled data with a large amount of unlabeled data during training, and unsupervised learning does not use labels at all during training.

2.4. Deep Learning

Deep learning refers to machine learning models that are based on *neural networks*. Neural networks have its name since it vaguely resembles the human brains, but it is simply a combination of linear transformations with a non-linear activation function.

Recurrent neural networks are a form of neural networks for processing sequential data. The recurrent neural networks cell takes two inputs; one value of the sequential data and a previous internal state. This internal state gives makes the model capable of remembering previous input and acts as a kind of memory.

The Long Short-Term Memory (LSTM) is an extension of the regular recurrent neural networks which is better at learning long-range dependencies. Due to the *vanishing gradient problem* during backpropagation, a regular recurrent neural network is not able to learn long-range dependencies. An LSTM network mitigates this problem by adding a forget gate, an input gate. and an output gate. These gates make the network able to choose how much of the previous state it will remember, what in the new input it will pay attention to and how much it should store for future use. These gates enable this model to learn longer-range dependencies. An overview of the LSTM architecture can be seen in Figure 2.2. LSTM was introduced by Hochreiter and Schmidhuber (1997).

A bidirectional Long Short-Term Memory (BiLSTM) is a method to give an LSTM more context on the future. In a vanilla LSTM, you will give the data to the LSTM sequentially from the past to the future. With a BiLSTM you will run your inputs in two ways, one from past to future and one from future to past. The output from these runs will be concatenated for future processing.

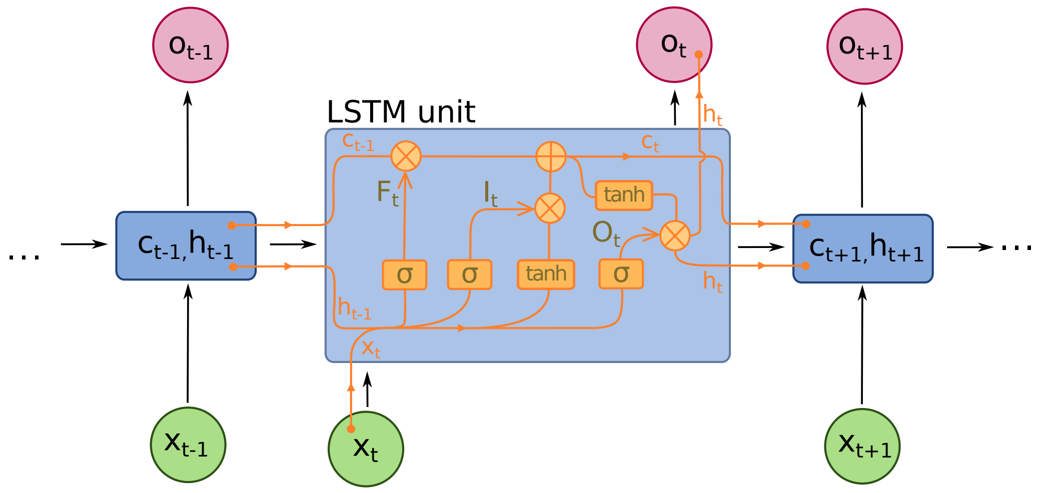

Figure 2.2.: A Long Short-Term Memory model. Figure from fdeloche, CC BY-SA 4.0, via Wikimedia Commons.

2.5. Evaluation

After an AMT system has made a prediction, we need a score to evaluate the generated transcription to a ground-truth transcription.

2.5.1. Precision, recall and F1-score

Precision, recall, and F1-score are evaluation metrics that are ubiquitous in the field of machine learning. These metrics are also used in the AMT evaluation and are presented here for reference.

$$Precision = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FP(t)} \tag{2.2}$$

$$Recall = \frac{\sum_{t=1}^{T} TP(t)}{\sum_{t=1}^{T} TP(t) + FN(t)} \tag{2.3}$$

$$F1 = \frac{2 \cdot Precision \cdot Recall}{Precision + Recall} \tag{2.4}$$

TP, *FP*, and *FN* are short for *true-positive*, *false-positive*, and *false-negative* respectively. In this case, these values are dependent on the time index t in the musical piece.

2.5.2. Frame-level evaluation

In the frame-level evaluation, we are only interested in the fundamental frequencies (F0) in a given time interval. We do not give any relevance to when a note begins, the onset, and if the frequency is coherent over the entire note duration in the ground-truth. This is a metric that is straightforward to implement and has no subjective hyper-parameters, however, it lacks the very audible idea of note onsets and consistency over time.

We define *true-positives*, *true-negatives*, and *false-negatives* as in Bay et al. (2009). *True-positives*, $TP(t)$, are defined as the number of F0s that correctly correspond in the prediction and the ground-truth. *False-positives* are the number of predicted F0s that are predicted, but not in the ground-truth. *False-negative* is the number of active fundamental frequencies in the ground-truth that are not in the prediction. To calculate the frame-level metrics equations 2.2, 2.3, 2.4 are used.

2.5.3. Note-level evaluation

Note-level evaluation is a higher-level evaluation than the frame-level evaluation. Instead of counting each frame independently, we look at the notes, more specifically the note onset and offset. Two evaluation methods are suggested in Bay et al. (2009), one that only accounts for the note onsets and one that also includes the note offset. In both cases, we define a note onset to be a true positive if the fundamental frequency is within a quarter note and the onset is within a given threshold. A common threshold is 50ms.

In the last case, the predicted note offset is required to be 20% of the ground-truth note's duration. As in section 2.5.2, *true-positives* are defined as those notes that conform to the previously mentioned requirements and the *false-positives* are defined as the ones that do not. *False-negative* is again the number of active fundamental frequencies in the ground-truth that are not in the prediction.

2.5.4. Note-level evaluation with velocity

Introduced in Hawthorne et al. (2018), note-level evaluation with velocity is an extension to the previous evaluation. Unlike the previous evaluations, velocity has no absolute meaning. As such, this evaluation is invariant under a constant scaling of velocities. To calculate this metric, the velocities in the ground-truth are first scaled to be in a range between 0 and 1 based on the overall highest velocity. After this, note pairs in the ground-truth and estimation are matched based on the pitch and onsets/offsets timing. A scaling factor is calculated based on minimizing the square difference on these pairs. All the note pairs with velocities within a given threshold are regarded as *true-positives*.

3. AMT Datasets

This chapter contains a list of datasets that have been used for automatic music transcription (AMT) in related work and descriptions of them. It also contains datasets that originally are created for other music informational retrieval fields, such as audio source separation, but can be useful for AMT. All of the datasets, except MUSDB18, uses the file format Musical Instrument Digital Interface (MIDI) for storing annotations.

The first four datasets, MAPS, MAESTRO, Expanded Groove MIDI Dataset, and MusicNet, are specifically created for automatic music transcription and consist of piano, piano, drums, and various classical instruments respectively. Million Song Dataset is a database of audio features from one million contemporary songs and contains preview clips for almost every song. The Lakh MIDI dataset is a large collection of MIDI files, and some of the MIDI files have been aligned to preview clips from the Million Song Dataset. Together these datasets can be used for supervised automatic music transcription. SLAKH consists of a subset of Lakh and is synthesized using professional-grade sample-based virtual instruments. MUSDB18 is created for audio source separation but can be used for unsupervised or semi-supervised AMT models in a stream-level transcription (for drums, bass, or vocal).

This section is equal to section 3 in Grønbech (2020) without any modifications.

3.1. MAPS

MAPS, an acronym for MIDI Aligned Piano Sounds, is a dataset of MIDI-annotated piano recordings created by Emiya et al. (2010). This is the oldest dataset specifically created for AMT on this list. The dataset consists of both synthesized piano audio and recordings of a Yamaha Disklavier piano. The Yamaha Disklaviers are concert-quality acoustic grand pianos that utilize an integrated high-precision MIDI capture and playback system. The MIDI capture system makes this piano series ideal to generate audio from a ground truth MIDI file.

The dataset is around 65 hours in total length and consists of four parts; one part consists of isolated notes and monophonic excerpts (ISOL), one of chords with random pitch notes (RAND), another of "usual chords" from Western music (UCHO), and the last of pieces of piano music (MUS). MUS consists of several recordings conditions with 30 compositions in each part. Two parts are performed by the Disklavier MIDI playback system.

3.2. MAESTRO

MAESTRO (MIDI and Audio Edited for Synchronous TRacks and Organization) is a dataset composed of over 172 hours of piano performances introduced by Hawthorne et al. (2019), the Magenta team at Google. The raw data is created from nine years of the International Piano-e-Competition events. In the competitions, virtuoso pianists performed on the Yamaha Disklavier piano, as was also done in section 3.1, but this time the integrated MIDI system was used to capture the performance. In the competitions this allowed judges to listen to the performances remotely on another Disklavier.

The MIDI data includes sustain pedal positions and key strike velocities. According to Hawthorne et al. (2019), the MIDI files are alignment with 3ms accuracy. As in section 3.1, the audio and MIDI files are sliced into individual musical pieces. There is also a suggested training, validation and test split such that the same composition does not appear in multiple subsets and the proportions make up roughly 80/10/10 percent. This dataset is around one order of magnitude larger than MAPS.

3.3. Expanded Groove MIDI Dataset

Expanded Groove MIDI Dataset is a dataset of MIDI-annotated drum performances. It was first released as Groove MIDI Dataset (Gillick et al., 2019) and later as an Expanded Groove MIDI Dataset (Callender et al., 2020). As the MAESTRO dataset, this comes from the Magenta team at Google. The dataset consists of 444 hours of audio from 43 drum kits. The MIDI data comes from 10 drummers, mostly hired professionals, performed on a Roland TD-11 electronic drum kit. The drummers were instructed to play a mixture of long sequences of several minutes of continuous playing and short beats and fills. All the performances were played with a metronome. This resulted in around 13.6 hours of human-performed MIDI files.

In Callender et al. (2020) the MIDI data was re-recorded on a Roland TD-17 drum module with 43 different drum kits, with both acoustic and electric drums. The recordings were done at 44.1 kHz sample rate and 24-bit audio bid depth and aligned within 2ms of the original MIDI files.

3.4. MusicNet

MusicNet (Thickstun et al., 2017) is a large-scale dataset consisting of 330 freely-licensed classical pieces. It is 34 hours in length and has a total of 11 instruments (piano, violin, cello, viola, clarinet, bassoon, horn, oboe, flute bass, and harpsichord). The pieces are composed by 10 different classical composers such as Beethoven, Schubert, Brahms, Mozart, and Bach. Various types of ensembles are recorded such as solo piano, string quartet, accompanied cello, and wind quintet. The labels are acquired from musical scores aligned to recordings by *dynamic time warping* and later verified by trained musicians. Thickstun et al. (2017) estimate a labeling error rate of 4%. However, Hawthorne et al. (2018) point out that the onsets alignments are not very accurate.

3.5. Million Song Dataset

The Million Song Dataset is a collection of audio features and metadata for a million contemporary popular music tracks (Bertin-Mahieux et al., 2011) The dataset includes information such as artist, artist location, tempo, key, time signature, year, and 7digital track ID. 7digital [1] is a content provider and Schindler et al. (2012) were able to download 994,960 audio samples from their service. Most of these samples were preview clips of length 30 or 60 seconds. While copyright prevents re-distribution of the audio snippets, other researchers have obtained the audio for future research (Raffel, 2016).

3.6. Lakh MIDI Dataset

Lakh MIDI dataset (LMD) is a collection of 176,581 unique MIDI files published by Raffel (2016). A subset of these files, 45,129, has been matched with the Million Song Dataset (see section 3.5). The MIDI files were scraped from publicly-available sources online and de-duplicated based on their MD5 checksum. The Lakh MIDI Dataset is distributed with a CC-BY 4.0 license at this website [2].

In the doctoral thesis (Raffel, 2016), several metrics on the MIDI files were calculated and some of the files were aligned to the preview clips in the Million Song Dataset. *Dynamic time warping* and other techniques were used to align the files. The word lakh is a unit of measure used in the Indian number system for the number 100,000. This word was used since the number of MIDI files were roughly of this order and it is a play on the Million Song Dataset.

3.7. SLAKH

The Synthesized Lakh (Slakh) Dataset, as the name implies, is a subset of the Lakh MIDI Dataset which has been synthesized using professional-grade sample-based virtual instruments. The dataset is created by Manilow et al. (2019) and is originally intended for audio source separation. A subset of 2100 MIDI files that contained the instruments *piano*, *bass*, *guitar* and *drums*, where each of these four instruments plays at least 50 notes, was chosen from LMD. The MIDI files have a suggested train/validation/test split of size 1500, 375, and 225 tracks, respectively.

Since this dataset was created for audio source separation, each of the individual instruments has been rendered. To add variety Slakh uses 187 patches categorized into 34 classes in which each of the instruments has been randomly assigned. Some of the patches are rendered with effects such as reverb, EQ, and compression. Some of the MIDI program numbers that are sparsely represented in LMD, such as those under the “Sound Effect” header, are omitted.

The first release of this dataset, unfortunately, had some compositions that occurred in more than one split. To not leak information between the splits two rearrange-

[1] `https://www.7digital.com`
[2] `https://colinraffel.com/projects/lmd`

ments have been suggested by Manilow et al. (2019), namely `Slakh2100-redux` and `Slakh2100-split2`. `Slakh2100-redux` omits tracks such that each MIDI files only occurs once, while `Slakh2100-split2` moves tracks such that no tracks occur in more than one split.

3.8. MUSDB18

MUSDB18, like Slakh, is a dataset created for audio source separation and accordingly does not contain annotations in the form of MIDI or other formats. It contains 150 full length human-created music tracks along with their isolated *drums*, *bass*, *vocal* and *others stems*. The tracks come from different sources, 100 tracks are taken from the *DSD100 dataset* (Ono et al., 2015; Liutkus et al., 2017), 46 tracks are from the Medley DB dataset (Bittner et al., 2014), and other material. The tracks have different licenses, some are under the Creative Commons license while a large portion of the tracks is restricted. The authors, Rafii et al. (2017), have suggested a training split of 100 songs and a test split of 50 songs.

4. Related Work

This chapter covers the related work that has been done on automatic music transcription and music source separation. In the first section, different approaches to automatic music transcriptions that have been applied through the years are presented to give an overview of the different possibilities and solutions that exist. The second section covers more recent approaches that use neural networks and these represent the state-of-the-art solutions to date. The third section covers briefly the work that has been done on music source separation.

Section 4.1 is equal to Section 4.1 in Grønbech (2020) with minor adaptions. Section 4.2 is based on Section 4.2 in Grønbech (2020).

4.1. Different Approaches to Multi-Pitch Estimation

This section covers the different approaches that have been implemented for multi-pitch estimation (also called multiple-F0 estimation) through the years, excluding the neural network approaches, which are presented in the next section. Multi-pitch estimation is a subtask of automatic music transcription that focuses on frame-level transcription. These approaches can be divided into three different general methods, according to Benetos et al. (2013). These three methods are:

- Feature-based multi-pitch detection
- Statistical model-based multi-pitch detection
- Spectrogram factorization-based multi-pitch detection

4.1.1. Feature-based multi-pitch detection

Most feature-based multi-pitch detection algorithms make use of methods derived from traditional signal processing. Fundamental frequencies are detected using audio features from the time-frequency representation. A *pitch salience function* (also called *pitch strength function*) that estimates the most prominent fundamental frequencies in a given audio frame is used. The pitch salience function usually emphasizes the magnitude of the partials (fundamental frequencies and overtones) in a power spectrogram. In Benetos and Dixon (2011) the partials are modeled as:

$$f_{p,h} = h f_{p,1} \sqrt{q + (h^2 - 1)\beta_p}, \tag{4.1}$$

where $f_{p,1}$ is the fundamental frequency (F0), β_p is an inharmonicity constant and $h > 1$ is the partial index. Inharmonicity can occur due to factors such as string stiffness, where all partials will have a frequency that is higher than their expected harmonic value. Given the frequencies of the overtones in equation 4.1, a salient function can be formulated. Since each partial can be shifted slightly given the instrument and pitch, a grid search for β_p and the number of partials to look for can be done on a labeled excerpt of the instruments that will be transcribed.

After a salient function is modeled, a *pitch candidate selection step* that selects zero, one, or several pitches is needed. A difficult aspect is that the partials of different fundamental frequencies may overlap and that octaves above the F0 often have a high value in the pitch salient function. One method to mitigate this is to iteratively choose the most prominent F0-frequency and remove its partials from the frequency domain until there are no more prominent pitches. More intricate selection steps that jointly choose F0-pitches have also been suggested (Benetos and Dixon, 2011).

Since pitch salient features solely based on the time-frequency representation are prone to octave-above errors, Su and Yang (2015) incorporate the *lag* (a.k.a. quefrency) domain, such as the autocorrelation function and logarithm cepstrum. They call this the *temporal representation* $V(\tau)$ where τ can be mapped to the frequency domain via the relationship $f = 1/\tau$. Since the partials are approximately multiples of the F0 in the time-frequency domain, it will be the inverse multiple in the temporal representation. Estimates in the temporal representation are prone to sub-octave errors, hence combining these features has increased performance in both the single-pitch estimation and multi-pitch estimation.

4.1.2. Statistical model-based multi-pitch detection

Multiple-F0 estimation can also be modeled in a statistical framework. Given a set of all possible F0 combinations $\mathcal{C}$ and an audio frame $\boldsymbol{x}$, the frame-based multiple-pitch estimation problem can be viewed as a maximum a posteriori (MAP) estimation problem (Emiya et al., 2010).

$$\hat{\mathcal{C}}_{MAP} = \operatorname*{arg\,max}_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} P(\mathcal{C}|\boldsymbol{x}) = \operatorname*{arg\,max}_{\mathcal{C}\in\boldsymbol{\mathcal{C}}} \frac{P(\boldsymbol{x}|\mathcal{C})P(\mathcal{C})}{P(\boldsymbol{x})} \tag{4.2}$$

where $\mathcal{C} = \{F_0^1, ..., F_0^N\}$ is a set of fundamental frequencies, $\boldsymbol{\mathcal{C}}$ is the set of all possible fundamental frequencies combinations, and $\boldsymbol{x}$ is the audio frame.

4.1.3. Spectrogram factorization-based multi-pitch detection

Before the rise of deep learning, the spectrogram factorization-based approach *non-negative matrix factorization* achieved the best results in multi-pitch estimation. First introduced in Smaragdis and Brown (2003), non-negative matrix factorization tries to factorize a non-negative input spectrogram, $\boldsymbol{V}$, into two matrices called a *dictionary*, $\boldsymbol{D}$, and an *activation* matrix $\boldsymbol{A}$.

$$\boldsymbol{V} = \boldsymbol{D}\boldsymbol{A} \tag{4.3}$$

This has an intuitive interpretation; matrix $\boldsymbol{D}$ contains the frequency information of the input spectrogram while the activation matrix contains the temporal information of when each note is active. If the frequencies for each pitch were stationary it would be possible to decompose $\boldsymbol{V}$ as in equation 4.3. As this usually is not the case, the goal is to minimize the distance between $\boldsymbol{V}$ and $\boldsymbol{DA}$. Smaragdis and Brown (2003) derived rules which can be used to update $\boldsymbol{D}$ and $\boldsymbol{A}$ to minimize this distance.

4.2. Automatic Music Transcription with Neural Networks

This section covers the state-of-the-art approaches for automatic music transcription (AMT) that have been published mostly in the last decade. In this period, neural networks have proven to be powerful for automatic music transcription, and have become increasingly popular. However compared to other fields such as image processing, progress on neural networks for automatic music transcription has been slower. All approaches and systems in this section include the use of neural models. Since neural network approaches have shown an increased performance, most of these models operate on the note-level transcriptions and not solely on multi-pitch estimation.

One of the first attempts at AMT with neural networks was originally published in 2001 with the work of Marolt (2004). Their model consists of adaptive oscillators that track partials in the time-frequency domain and a neural network that is used for note-prediction. Different neural network architectures were tested such as multilayer perceptrons, recurrent neural networks, and time-delay neural networks. The time-delay neural gave the best results in their experiments.

One of the first successful approaches in more recent years is the system presented by Böck and Schedl (2012). This system uses two parallel *Short-Time Fourier Transforms* (STFT) with different window lengths to capture both a high temporal and frequency representation. The idea is that the high temporal resolution helps to detect note-onsets and the increased frequency resolution makes it easier for the model to disentangle notes in the lower frequency range. The magnitudes of the two spectrograms are then log-scaled and aligned according to the pitches of the 88 MIDI notes. This representation is used as an input to a bidirectional Long Short-Term Memory (BiLSTM) recurrent neural network. The output of the network is the predicted note onsets for each of the 88 pitches. The note offsets or note durations are not predicted.

Later work has been inspired by speech recognition by extending an acoustic front-end with a symbolic-level module resembling a language model (Sigtia et al., 2016). The symbolic-level module is called a *music language model* (MLM) and the rationale with this module is to learn long-range dependencies such as common chord progressions and melodic structures. This module uses a recurrent neural network to predict active notes in the next time frame given the past and can be pre-trained on MIDI files independently of the acoustic model. The MLM improved performance in all evaluations in their work, however, the increase was modest (around one point in F_1-score).

Sigtia et al. (2016) were the first to use convolutional networks in their acoustic model. There are several motivations for using a convolutional network for acoustic modeling.

Previous work suggests that rather than classifying a single frame of input, better prediction accuracies can be achieved by incorporating information over several frames of inputs. This has typically been achieved by applying a context window around the input frame or by aggregating information over time by calculating statistical moments over a window of frames. Applying a context window around low-level features such as STFT frames, however, would lead to a very high dimensional input which can be computationally infeasible. Also, taking the mean, standard deviation or other statistical moments makes very simplistic assumptions about the distribution of data over time in neighbouring frames. Due to their architecture, convolutional layers are directly applied to neighboring features both in frequency and time dimensions. Also, due to their weight sharing, pitch-invariant features in the log-frequency representation can be learned.

Kelz et al. (2016) disregarded the music level model completely and did a comprehensive study on what could be achieved only with an acoustic model. They also studied how different input representations affect performance. In their work, they only focused on frame-level transcriptions. Four different spectrograms were examined; spectrograms with linearly spaced bins, spectrograms with logarithmically spaced bins (mel-scaled), spectrograms with logarithmically spaced bins and logarithmically scaled magnitude, as well as the constant-Q transform. For a shallow neural network, Kelz et al. (2016) achieved the best performance with a spectrogram with logarithmically spaced bins and logarithmically scaled magnitude. Hence a mel-scaled spectrogram with 229 bins and logarithmically scaled magnitude is a common choice in more recent work (Hawthorne et al., 2018; Kong et al., 2020). Models with convolutional layers outperform dense neural networks, but whether or not the convolutional layers are followed by a dense layer was not as significant for performance. Kelz et al. (2016) achieved a frame-level F_1-score of around 70 on the Yamaha Disklavier piano recordings in MAPS (see section 3.1).

The next leap in performance was done by Google's Magenta[1] team with the model Onsets and Frames (Hawthorne et al., 2018). With this model, they focus on note-level transcriptions of pianos. Since the piano has a perceptually distinct attack and the note-energy decays immediately after the onset, Hawthorne et al. (2018) argue that these frames should be given special relevance. The model is split into two parts, one part detects note onsets and the other part detects fundamental frequencies in frames conditioned by the onsets. The onsets are further emphasized by giving frame labels closer to the onsets a higher value. The onsets detection head has several advantages, predicting note onsets is easier than predicting fundamental frequencies of independents frames, hence this mitigates the common problem of spurious predictions. Secondly, since active frames is a much more common event, conditioning it on onsets also reduces the number of false positives. This model architecture could, however, struggle to predict instruments that do not have as prominent note onsets such as string and wind instruments when starting the note softly. The architecture in the original model is shown in Figure 4.1. Both the frame and onset detection head are trained jointly. Hawthorne et al. (2018) achieved a note F_1-score of 82.3 and a note-with-offset score of 50.2 on the MAPS Disklavier piano recordings. This is also the first model to predict note onset velocity.

[1] https://magenta.tensorflow.org

This is done with a similar stack as the frame and onset model but is not conditioned on the others.

In a later revision, the team from Magenta achieved much higher performance with an extended Onset and Frames model. In this model Hawthorne et al. (2019) added an offset detection head, inspired by Kelz et al. (2019). This offset stack is not used in inference but fed to the frame stack together with the onset stack. Hawthorne et al. (2019) also simplified the frame value for the loss function by not decaying the weights. The number of parameters in the new model is also increased significantly and together with the much larger training dataset, MAESTRO (see section 3.2), this model achieved a note F_1-score of 86.44 and a note-with-offset F_1-score of 67.4 on the MAPS Disklavier piano recordings. On the MAESTRO test dataset, this model achieved a 95.32 note F_1-score and 80.5 note-with-offset F_1 score.

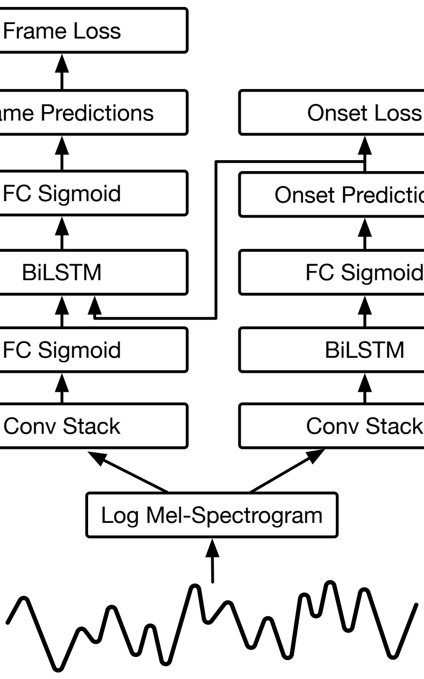

Figure 4.1.: Architecture of the original Onsets and Frames model. Reprint of Figure 1 from Hawthorne et al. (2018) under CC BY 4.0 license.

Kim and Bello (2019) used the extended Onsets and Frames model as a reference and added an adversarial training scheme. They point out that many of the current state-of-the-art transcription models, such as the one from Hawthorne et al. (2018), use an element-wise sum of the pitch labels and prediction as a loss function which indicates that each element is conditionally independent of each other. This encourages the model to predict the average of the posterior, which can also be seen in image tasks with similar loss functions resulting in blurry images. Kim and Bello (2019) added a discriminator to the loss function inspired by GANs' success on image translation tasks.

The computation graph of the loss function is shown in Figure 4.2. The authors point out that the discriminator effectively implements a music language model (MLM) and biases the transcription towards more realistic note sequences. They achieve a note F_1-score of 95.6 and a note with an offset F_1-score of 81.3 on the MAESTRO test dataset.

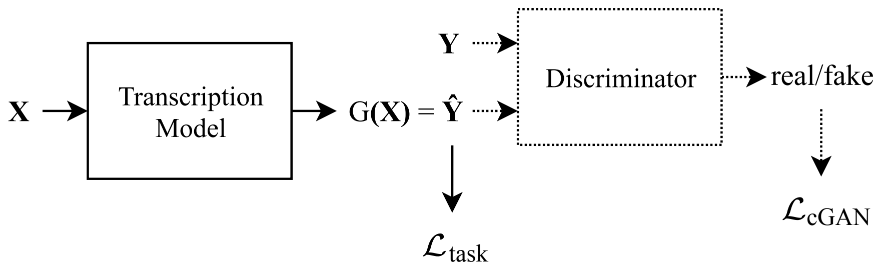

Figure 4.2.: Computation graph of adversarial loss in Kim and Bello (2019). Reprint of Figure 2 from Kim and Bello (2019) under CC BY 4.0 license.

More recently, a team from ByteDance[2] Kong et al. (2020) has achieved an ever better score on the MAESTRO test dataset. Their model, just as the extended Onsets and Frames model, uses an onset, offset, frame and velocity stack. One of the new ideas in the model architecture is that the onset stack is conditioned on the velocity stack. Just as in the Extended Onsets and Frames model, the frame stack is conditioned on both the onset and offset stack in addition to an activation stack. Kong et al. (2020) also analytically calculate onsets and offsets instead of rounding them to the nearest frame. Together these changes increase note-level F_1-score to 96.72 and a note-with-offset F_1 score 82.47 on the MAESTRO test dataset. This model also incorporates sustain pedal prediction and achieves an F_1-score of 91.86 on the same training set.

Hung et al. (2020) focus on stream-level transcription and use the dataset Slakh (see section 3.7) for both training and evaluation. Their model performs music source separation and transcription jointly using an adversarial approach. A separator, that acts as a generator, outputs a time-frequency mask for each instrument, and a transcriptor that acts as a critic, provides both temporal and frequency supervision to guide the learning of the separator. The authors write that they achieve a transcription note accuracy on `Slakh2100-split2` of 86% on bass, 51% on guitar, and 61% on piano on the mixture tracks.

Results for the most recent work in automatic music transcription are shown in Table 4.1. Rows above the horizontal line are on the MAPS Disklavier piano recordings and below on the the MAESTRO test split (see section 3.1 and 3.2). Since Sigtia et al. (2016) and Kelz et al. (2016) did not calculate all the metrics in their paper, the reproduced results from Hawthorne et al. (2018) are used. We notice that Hawthorne et al. (2018) achieved significantly higher score that the prior work with the Onset and Frames architecture on the MAPS dataset. We also see that Hawthorne et al. (2019) got much higher results on the MAESTRO dataset than on the MAPS dataset with the

[2] https://www.bytedance.com

Table 4.1.: Automatic music transcription results on piano. Rows above the horizontal line are evaluated on MAPS and below on MAESTRO

	Frame			**Note**			**Note w/offset**		
	P	R	F_1	P	R	F_1	P	R	F_1
Sigtia et al. (2016)	72.0	73.3	72.2	45.0	49.6	46.6	17.6	19.7	18.4
Kelz et al. (2016)	81.2	65.1	71.6	44.3	61.3	50.9	20.1	27.8	23.1
Hawthorne et al. (2018)	88.5	70.9	78.3	84.2	80.7	82.3	51.3	49.3	50.2
Hawthorne et al. (2019)	92.9	78.5	84.9	87.5	85.6	86.4	68.2	66.8	67.4
Hawthorne et al. (2019)	92.1	88.4	90.2	98.3	92.6	95.3	83.0	78.2	80.5
Kim and Bello (2019)	93.1	89.8	91.4	98.1	93.2	95.6	83.5	79.3	81.3
Kong et al. (2020)	88.7	90.7	89.6	98.2	95.4	96.7	83.7	81.3	82.4

same model. A possible explanation for this is that the labels for the MAPS dataset are not fully accurate. Another trend is that the more recent work achieves better F_1 scores and that the precision is higher that the recall, except for the frame results in Kong et al. (2020) which are lower than the previous works. Kong et al. (2020) did reproduced the work in Hawthorne et al. (2019) with slightly worse score.

4.3. Music Source Separation

This section briefly covers the state-of-the-art approaches for music source separation that have been published in the last couple of years. All of the presented models are deep learning-based, and since deep learning models are inherently dependent on learning data, deep learning models first kicked off after the large-scale publicly available dataset MUSDB18 (see Section 3.8) was released in 2017.

Open-Unmix by Stöter et al. (2019a) serves as an reference implementation for deep learning-based music source separation and is from the same group[3] that created MUSDB18. Their work aimed to achieve state-of-the-art performance and be easily understandable so it could serve as a basis for other researchers in the future. Stöter et al. specifically state that Open-Unmix should be 'MNIST-like' and as such, is openly available on GitHub[4] and is encouraged to be hacked on by other researchers. The main idea in the model architecture is as follows; we know that spectrograms (see Section 2.2.2) can be losslessly reversed back to the original waveform signal by the *inverse discrete Fourier transform*. What would happen if we changed the magnitude of the spectrogram before doing the inverse transformation? It happens that this setup works well for music source separation and is what the architecture does. It takes a magnitude-based spectrogram as input, uses three BiLSTMs with a skip-connection to predict a mask, and multiplies this mask with the original spectrogram. A separate model is used for each instrument and the model is optimized in the magnitude domain using mean squared error.

[3]`https://sigsep.github.io`
[4]`https://github.com/sigsep/open-unmix-pytorch`

Another model was created by a group from Deezer[5] called Spleeter (Hennequin et al., 2020). Spleeter also operates on the magnitude spectrogram domain but the model is changed to a U-Net architecture. The U-Net architecture is a encoder-decoder kind of architecture with skip-connections between all the layers of equal size. Both the downsampling and upsampling-block consist of 2D convolutional layers. Since the architecture in solely based convolutional layers, it is very efficient. It is able to separate three and a half hours of audio into four stems in less that two minutes on GPU while competing with state-of-the-art results on the MUSDB18 dataset.

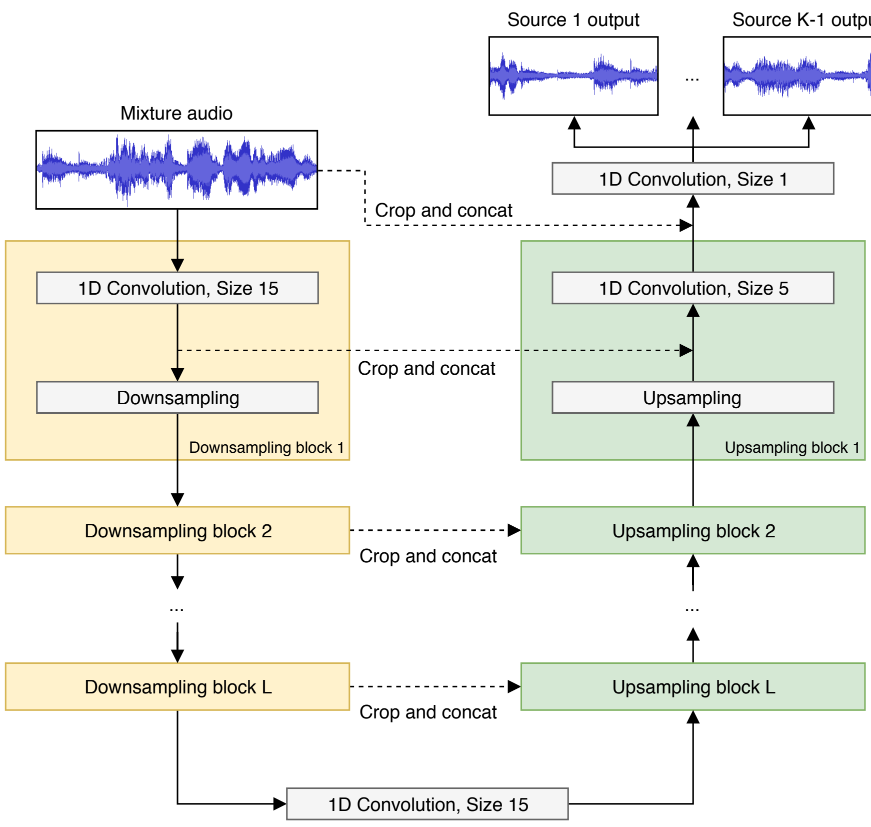

Figure 4.3.: Wave-U-Net architecture. Reprint of Figure 1 from Stoller et al. (2018) under CC BY 4.0 license.

Wave-U-Net is by Stoller et al. (2018) and operates directly on the waveform domain. Most of the earlier work operates on the magnitude spectrogram, which ignores phase information and relies on a fixed set of hyperparameters for the spectrogram. Stoller et al. created an adaption of the U-Net architecture to the 1-dimensional input dimension. By down-sampling the input features L times, the model operates on a longer and longer temporal window. As we see in Figure 4.3, there are also skip-connections between the downsampling and upsampling-block to propagate high-frequency information. The

[5]https://www.deezer.com

output layer is engineered to enforce source additivity to the K separated instruments.

CatNet is a more recent work by Bytedance that operates both in the time-frequency domain and the raw waveform domain (Song et al., 2021). The framework concatenates a U-Net separation branch using spectrogram as input and a Wav-U-Net separation branch using time-domain waveform as input for music source separation. By combining both representations the model can incorporate phase information and the frequency information from a spectrogram. With this architecture state-of-the-art results were achieved on vocal separation.

5. Architecture

The following chapter describes the model architectures and post-processing that will be used in the experiments in chapter 6. First, the extended Onsets and Frames architecture will be described. This architecture with minor variations has been used in several recent papers in a single-instrument setting as described in section 4.2. The next section describes the new multi-instrument automatic music transcription architecture that combines the extended Onsets and Frames model with a U-Net architecture. The U-Net architecture is inspired by the architectures that are typically used in Music Source Separation (see section 4.3). Finally, the post-processing step to extract notes from the piano-roll representation for automatic music transcription is shown.

Section 5.1 is based on Chapter 5 in Grønbech (2020).

5.1. Extended Onsets and Frames

The architecture used in most of the experiments in chapter 6 is shown in Figure 5.1. This architecture was created by Hawthorne et al. (2019) and based on their previous work in Hawthorne et al. (2018). The raw audio waveform is transformed into the time-frequency representation by a mel-scaled spectrogram with logarithmically scaled magnitudes as suggested by Kelz et al. (2016). Four detectors are then trained simultaneously to predict note onsets, frame activation, note offsets, and velocity. Each of the detectors has the same architecture except the frame prediction detector that does not have the convolutional stack. The convolutional stack works as an acoustic model and consists of three 2D convolutional layers with padding 1 and kernel size 3. Each convolutional layer is followed by a batch normalization layer and ReLU activation function. The last convolutional layer has a max-pooling operation to reduce the number of dimensions by half and a dropout of 0.25. After the last convolutional layer, a new max-pooling operation is added as well as a new dropout of 0.25. The convolutional stack is followed by a linear layer and a dropout of 0.5.

The convolutional stack is followed by a bidirectional LSTM, and a fully connected layer with sigmoid activation functions. The output from the onset prediction, frame activation prediction, and offset prediction is then fed into the final frame prediction model. The gradient propagation into the onset detector and offset detector from the frame network is stopped to make the training more stable. In practice, this means that the onset detector and offset detector are trained independently of the frame prediction detector. The frame model only consists of a bidirectional LSTM and a fully connected layer with sigmoid activation functions.

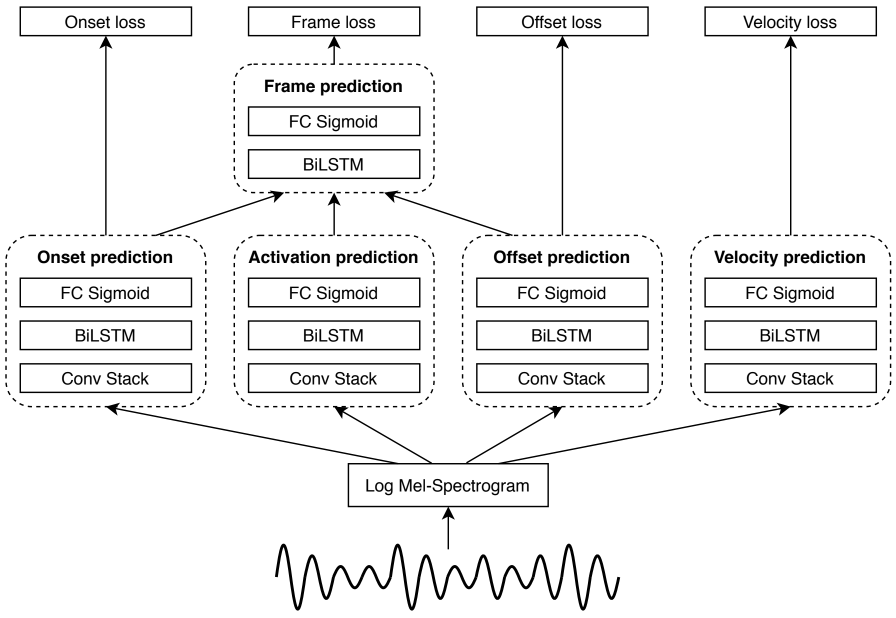

Figure 5.1.: The extended Onsets and Frames architecture. Reprint of Figure 5.1 in Grønbech (2020)

The loss function for this model is the sum of the loss function for the different detectors.

$$L = L_{onset} + L_{frame} + L_{offset+L_{velocity}} \tag{5.1}$$

L_{onset}, L_{frame} and L_{offset} are calculated as the binary cross-entropy between the respective prediction and label. The velocity loss is calculated as follows

$$L_{velocity} = \frac{\sum_{t=0}^{T} \sum_{p=p_{min}}^{p_{max}} I_{onset}(p,t)(V_{pred} - V_{label})^2}{\Sigma I_{onset}}, \tag{5.2}$$

where T is the number of frames, p_{min} and p_{max} are the lowest and highest pitch in the labels, I_{onset} is the onset label, and V_{pred} and V_{label} are the predicted velocity and velocity label, respectively. The motivation of multiplying the square of the difference of the velocities by the onset label is that we are only interested in the velocity at the note onset. The onset label is zero when the note is not active, hence the other velocities will not contribute to the loss function.

Due to the fact that there is no gradient propagation between the detectors, a weighting of the loss function in equation 5.1 would not influence the backpropagation.

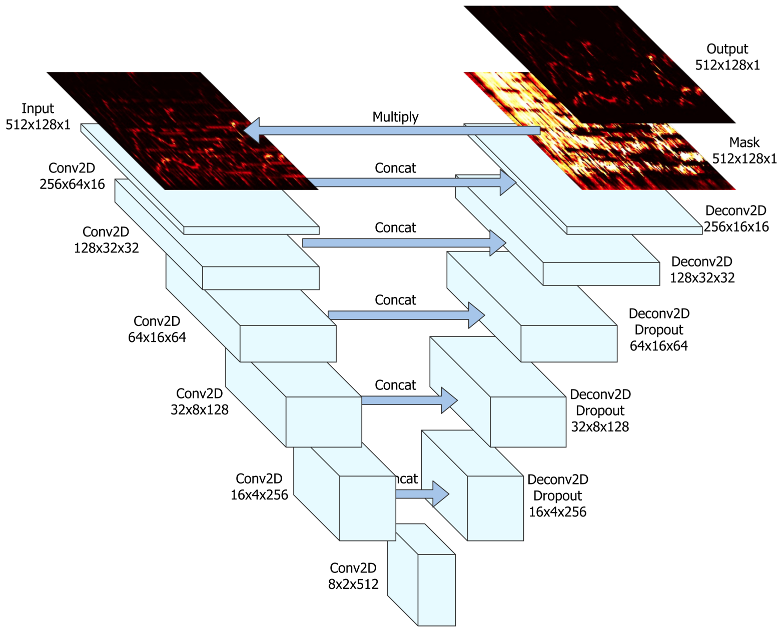

Figure 5.2.: The U-Net architecture. Reprint of Figure 1 in Jansson et al. (2017) under CC BY 4.0 license.

5.2. Extended Onsets and Frames with U-Net

This architecture aims to solve Research Question 3 by combining a music source separation model with an automatic music transcription model. A U-Net model is added between the log scaled mel-spectrogram and the Extended Onsets and Frames model from Section 5.1. The U-Net architecture is a kind of encoder-decoder architecture with additional skip-connections as can be seen in Figure 5.2. The left part in the Figure is the encoder and consists of six Conv2D blocks. Each Conv2D block consists of a 2-dimensional convolutional filter with kernel size 5, stride 2 and padding 2. Following the convolutional filter, a 2-dimensional batch normalization layer is added as well as a leaky-ReLU activation function. For each of the Conv2D blocks, the width and height are reduced in half, while the depth is doubled. This means that the number of features is reduced significantly and the encoder operates on a longer temporal and frequency window for each block. In the decoder, Deconv2D blocks are used. The Deconv2D block consists of transposed convolutions to upscale the width and height by a factor of two. Following the transposed convolution a ReLU activation function and a 2-dimensional batch normalization is added. The non-activated output of the corresponding encoder layers are concatenated to the previous Deconv2D block to propagate information more directly from the encoder to the decoder. This is shown as arrows marked with 'Concat' in the Figure. In the final layer, a Sigmoid activation function is used to truncate the values between 0 and 1 before it is multiplied by the input. This final layer works as a mask that only keeps the frequencies in the input that are relevant.

In Experiment 3 in Chapter 6 only the loss function for automatic music transcription is used (the separated audio source is not given as labels).

This is the same architecture used in Hennequin et al. (2020) and the implementation in this work is based on a re-implementation in PyTorch from this GitHub repository[1].

5.3. Post-Processing

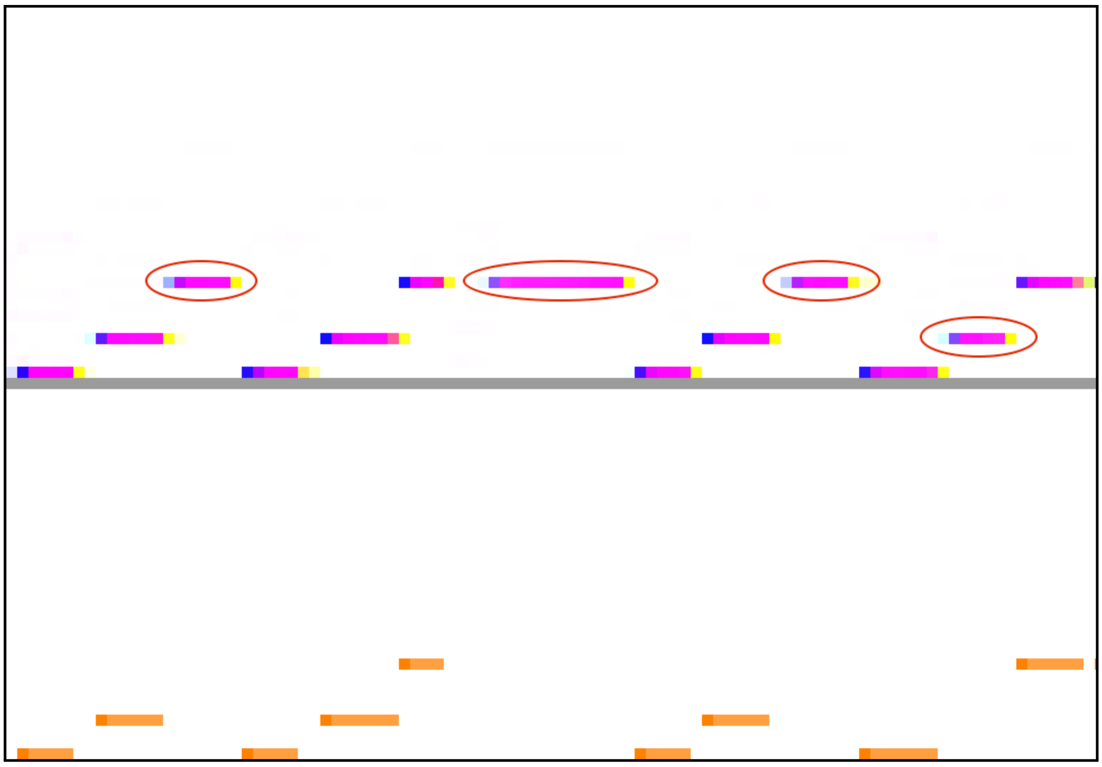

Figure 5.3.: Predictions and post-processed notes. In the upper half; blue represents onset, purple frame and yellow offset. Predictions that have not been processed to notes are marked with a red ring.

The Onsets and Frames architectures operate in a discrete piano-roll representation of music. To convert the piano-roll representation to MIDI files a post-processing step is needed. The typical post-processing step used in the related work is as follows; the onsets and frame prediction are binarized by a chosen frame threshold and onsets threshold. The offset prediction is neglected in this step. Every onset over the given threshold will be the start of a note. Each frame value above the chosen frame threshold is appended to the note length if it is at the same pitch and straight after a note onset or a chain of consecutive frames after the onsets. The frame and onset thresholds used in the related

[1]https://github.com/tuan3w/spleeter-pytorch

work are typically not specified, but a value of 0.5 for both is natural since the predictions will be between 0 and 1.

In Table 4.1 (page 25) the automatic music transcription results for the related work are shown. In that table, we see that the precision is higher than the recall in almost all the metrics. For some applications, higher recall is more important than higher precision, such as when using automatic music transcription models to compose music since it is normally easier to remove wrong notes than add ones that do not exist. Since the recall is lower, we want to increase the number of notes that get accepted without sacrificing the precision too much. One strategy to do this is to reduce the frame threshold and onset threshold.

The usual post-processing is shown in Figure 5.3 with onset and offset thresholds of 0.5. The prediction is shown in the upper half and the post-processed notes in the lower half. Blue represents onsets, purple frames and yellow offsets. Since the onset needs to be above a given threshold to start a note, not all frame predictions in the figure actually contribute to a processed note. The predictions marked with a red ring are in fact a correct prediction, but since the onset predictions are too weak it is not processed to a note. A modification to the usual post-processing step is proposed in this work. Since no information flows from the onset detector to frame detector in the Extended Onset and Frames architecture (see Figure 5.1), more relevance is given to the frame predictions. Instead of only using the predicted onset values to start a note, a note can start if there is a weak onset value and strong frame value straight afterward. This would increase the recall in the given figure. In the experiments in Chapter 6, both the usual and modified post-processing steps are tested on the validation set. The processing step with the best F_1 scores is chosen for each experiment. All the experiments also use an onset threshold and frame threshold lower than 0.5 to increase the recall.

6. Experiments and Results

As part of the research in this thesis, some experiments were carried out to answer questions related to the Research Questions and the overall goals of the thesis. This chapter presents the experiments conducted to investigate whether multi-instrument automatic music transcription is possible. Experiments 1–3 transcribe a single instrument from a multi-instrument audio source without any pre-processing, with audio source separation, and with a new architecture created for this purpose. Experiments 4–5 investigate if it is possible to train a note-level multi-instrument automatic transcription model and measure how well this model performs on different instrument classes. Experiment 6 introduces a stream-level multi-instrument model. This model is not only able to transcribe the audio source, but also tell which instrument a note is played by. The first section presents the overall plan for carrying out the series of experiments. The second section provides the reader with an overview of each experiment, together with the information necessary to reproduce the experiments. The final section presents detailed results obtained from each experiment.

6.1. Experimental Plan

To keep a series of experiments effective and in a structured fashion, an experimental plan is crucial. Seven different experiments are presented. Experiment 0 serves as an upper baseline for the other experiments since it operates in the simplified single-instrument automatic music transcription domain. Experiment 1, 2 and 3 are designed to answer Research Question 1, 2, and 3 respectively.

In experiment 4 a note-level multi-instrument model is trained. This model is evaluated in three different ways in experiment 4 and on twelve instrument classes in experiment 5.

Finally, in experiment 6 a stream-level multi-instrument model is trained. This model is not only able to transcribe pitched sounds, but also able to tell whether the note is played by piano, guitar, bass, a pitched instrument or drum. All the instruments and trained and inferred simultaneously.

Experiment 0 – Baseline Experiment

The first experiments will work as an upper bound for the other experiments. For this initial experiment, the Extended Onset and Frames model (see Section 5.1) will operate in a single-instrument automatic music transcription domain. In this experiment, the model will be trained and evaluated on electric bass, piano and guitar audio segments from the Slakh dataset.

6.1.1. Experiments on Pre-Processing and Model Architectures

Experiment 1–3 investigates the effect of pre-processing and different model architectures for automatic music transcription. Just as in Experiment 0, three of the most common instruments in the Slakh dataset are chosen–electric bass, piano and guitar. Electric bass is chosen instead of all the bass instruments since was done in Hung et al. (2020).

Experiment 1 – Without Source Separation

Experiment 1 is designed to directly answer Research Question 1. In this experiment, the same model as in Experiment 0 will be trained, however, this time the audio source will be multi-instrument (full band music, labeled as 'mix' in Table 6.3). In this setting, the model needs to filter out the instruments that are not relevant.

Experiment 2 – With Source Separation

Experiment 2 is designed to directly answer Research Question 2. The audio in this experiment will be pre-processed by a music source separation model. First, the model Spleeter by Hennequin et al. (2020) was used. This model, however, was not able to separate the bass from the Slakh music source at all. Subsequently, the model Open-Unmix by Stöter et al. (2019b) was used with much better results. Both models can separate music into four different stems—vocals, drums, bass and other separation. The audio source in this experiment will be the separated bass stems.

Experiment 3 – New Architecture

Experiment 3 is designed to directly answer Research Question 3. This experiment has the same setup as in Experiment 1 but a new architecture is created to both separate and transcribe the input audio (see Section 5.2). In this experiment, only the automatic music transcription labels will be used during training.

6.1.2. Experiments on Note-Level Multi-Instrument Transcription

Experiment 4 is designed to answer Research Question 4 by training a note-level multi-instrument model. This note-level model is created and trained in such as way that it should be able to transcribe all pitched notes but not be able to tell which instrument the note comes from. This model will be trained on the 'mix' audio source and be given the labels for MIDI stems with program numbers 0–95 (rest of the MIDI programs are either sound effects or very rare instruments). The model trained in this experiment will be evaluated in three different ways in Experiment 4 and on twelve instrument classes in Experiment 5.

Experiment 4a

In this experiment, the note-level multi-instrument model will be evaluated on the same kind of audio source the model was trained on ('mix').

Experiment 4b

In this experiment, the note-level multi-instrument model will be evaluated on all the instruments the model was trained on but this time the rest of the instruments in the audio source will be removed (such as drums and sound effect). This experiment together with experiment 4a will give some numbers on how much the effect of drums and other kinds of musical noise influences the transcribing performance.

Experiment 4c

In this experiment, the model is evaluated on the electric bass labels with the 'mix' audio source. This experiment is created to give a comparison to Experiment 1–3. Since this model is not able to tell which instrument played a specific note, only the recall will be presented. In other words, this experiment tries to answer – is the multi-instrument model able to transcribe as much of the electric bass as the specialised models in experiment 1–3?

Experiment 5 – Evaluation on twelve Instrument Classes

Experiment 5 is designed to directly answer Research Question 4. The note-level multi-instrument model trained in Experiment 4 will be evaluated on twelve different instrument classes. The instrument classes will be piano, chromatic percussion, organ, guitar, bass, string, ensemble, brass, reed, pipe, synth lead and synth pad. The audio source will be single-instrument—it will only consist of the instrument class it is evaluated on.

6.1.3. Experiments on Stream-Level Multi-Instrument Transcription

In this final experiment, a stream-level multi-instrument model is trained. This model will not only transcribe all the pitched sounds as in Experiment 4, but also tell if a note is played by bass, guitar, piano or drum. The model architecture is the Expended Onsets and Frames as discussed in Section 5.1, however, the labels for the different instruments are concatenated. The post-processing step is slightly modified the take the concatenated labels into account (but the algorithm described in Section 5.3 remains the same). This model will be trained on piano, guitar, bass, all the pitched instruments and drums simultaneously.

Experiment 6

The stream-level multi-instrument model is evaluated on the Slakh `redux` dataset. All the tracks that contain pitch-bends or do not have any notes played by at least one of the instrument classes are removed.

6.2. Experimental Setup

To make the results provided reproducible, the following section is dedicated to the details of the experimental setup used and the libraries and tools used for the experiments. The first codebase[1] is for the model architecture and contains all the experiments. The second codebase[2] is a PyTorch dataset for Slakh which enables fast and convenient loading of the audio and labels as well as necessary dataset modifications listed in section 6.2.1. Both of these codebases is based on the work by Kim (2019).

6.2.1. Dataset

The Slakh dataset (see section 3.7) by Manilow et al. (2019) is used in all the experiments. The dataset consists of 2100 tracks which are taken from the Lakh dataset (see section 3.6) (Raffel, 2016). Each track in Slakh consists of the following files; the original MIDI file from Lakh called `all_src.mid`, the mixed music called `mix.flac`, a file with metadata about the rendering called `metadata.yaml`, an individual audio track for each instrument which is saved in a folder called `stems` and a MIDI file for each stem in the `MIDI` folder. It is important to note that not all tracks in the original `all_src.mid` are rendered. Likewise, not all tracks in the `MIDI` folder are necessarily rendered either. To find out which tracks are rendered, the `metadata.yaml` file needs to be parsed.

For automatic music transcription, pitch bends should also be given special consideration. Of the 2100 tracks in the Slakh dataset, 1486 tracks contain at least one pitch bend that is a semi-tone above or below the note in the MIDI files. If we only look at the eletric bass stems, 930 tracks contain at least one pitch bend. Electric bass stems are MIDI tracks that have program numbers 33–37 (Electric Bass (finger), Electric Bass (pick), Fretless Bass, Slap Bass 1 or Slap Bass 2).

The original Slakh dataset contains several duplicated songs but where each track was rendered with different patches and effects. The duplication of tracks in different dataset splits (training, validation and testing) should be avoided in the automatic music transcription setting due to leakage between splits. This has been addressed by the Slakh authors, Manilow et al. (2019), by providing two new splits of the dataset called `redux`, `splits_v2` together with the `original` split. The `redux` split removes all the duplicated songs, while the `splits_v2` split moves tracks so that each track only occurs in one dataset split (training, validation or testing).

In the following experiments, the split that is used is the `redux` split but all the tracks that contain pitch bends are removed. For the electric bass experiments, all the tracks that contain more than two notes played simultaneously are also removed. This is done since some of the stems only play chords at high pitches and were probably given the instrument electric bass by a mistake. Some tracks also use a fretless bass to denote the melody and are played at a much higher pitch than a typical bass. Both of these cases are addressed by removing the stems that contain more than two bass notes played at the same time.

[1] https://github.com/greenbech/multi-instrument-onsets-and-frames
[2] https://github.com/greenbech/slakh-pytorch-dataset

Table 6.1.: Parameters for mel-scaled spectrogram. Experiment 0–2 use 229 frequency bins while experiment 3–4 256 frequency bins.

Sample rate	16kHz
Hop length	512
Window length	2048
Frequency bins	229/256
Min frequency	30Hz
Max frequency	8kHz

Another aspect of Slakh worth noticing is that Manilow et al. (2019) made a mistake when rendering and creating the MIDI files for bass stems. They truncated the MIDI pitches between MIDI note 35 which is a B1 (fundamental frequency 61.74hz, one octave above the deepest note on a 5-string electric bass) and MIDI note 79 which is G5 (fundamental frequency 783.99hz, one octave above the 24th fret on the highest string on a 4/5-string bass). That means that the seven deepest notes on a four-string bass and the deepest octave on a five-string bass are not present in the dataset. Likewise, the bass synth was not able to render the highest octave. That means that there are some pitches that are present in the labels that are not rendered in the audio. The PyTorch dataset created during the work in this thesis [3] takes this into account by removing the labels for the highest octave for bass. This dataset also removes tracks that were found to errors. A detailed list of stems with errors can be found in Appendix A.

6.2.2. Parameters

The parameters for the mel-scaled spectrogram are shown in Table 6.1. Experiment 0–2 use 229 frequency bins as this is commonly used in the related work. The U-Net model in experiment 3 requires the input features to be multiples of $2^6 = 62$, hence the number of frequency bins is increased to 256. Experiment 4 and 6 also use 256 frequency bins.

In the Extended Onsets and Frame model the convolutional layer has a kernel size of 3 and 1 padding. The number of filters for the convolutional layers are 48/48/96, the bidirectional LSTM unit size 384 and the fully connected layers have 768 units.

All the convolutional layers for the U-Net model have a kernel size of 5, stride of 2 and padding of 2. The number of convolutional layers are 16/32/64/128/256/512.

An onset threshold of 0.35 and frames threshold of 0.3 are chosen for most of the experiments after a sparse hyper-parameter search.

The number of trainable parameters for the different models are shown in Table 6.2.

An Adam optimizer with learning rate 0.0006 was chosen in all the experiments. The learning rate decays with a factor of 0.98 every 10 000 iteration. In addition to this learning schedule, an adaptive gradient clipping strategy called AutoClip by Seetharaman et al. (2020) was used. This clipping strategy restricts the gradient step by a given

[3] https://github.com/greenbech/slakh-pytorch-dataset

Table 6.2.: Trainable parameters for the different models

Experiment	**Parameters**
Ex. 0	21.6 MB
Ex. 1	21.6 MB
Ex. 2	21.6 MB
Ex. 3	32.8 MB
Ex. 4	29.6 MB
Ex. 6	38.2 MB

percentile (p) of the observed gradient norms during training. A p-value of 10 was chosen for all the experiments.

6.2.3. Environment and Resources

All the experiments were carried out on the NTNU HPC Idun Cluster (Själander et al., 2019). The Slakh PyTorch dataset is created in such a way that it can either stream from disc or keep all the labels and audio in memory. For faster training the audio should be kept in memory, but that also requires slightly above 16GB of RAM for the entire Slakh training and validation split. 16GB of VRAM is also required to run the models with the batch size and number of parameters in this thesis. All the experiments took around 8–11 hours to train on a NVIDIA V100/P100.

6.3. Experimental Results

This sections presents the results. All the results are calculated by the `mir_eval` Raffel et al. (2014) library implemented in Python. The average for all the tracks in the test set are reported. Inference is relatively fast for all models in the order of 5–10 times real-time performance on CPU and 10–100 times on GPU.

6.3.1. Experiment 0–3

Table 6.3.: Results for experiment 0–3 on the modified Slakh `redux` test dataset split

	Instrument class	**Audio source**	**Frame** P	R	F_1	**Note** P	R	F_1	**Note w/offset** P	R	F_1
Ex. 0a	Electric bass	Individual	99.4	97.9	98.5	99.8	98.6	99.1	98.6	97.4	97.9
Ex. 0b	Piano	Individual	90.7	89.5	89.6	97.5	96.3	96.9	72.2	71.3	71.7
Ex. 0c	Guitar	Individual	93.6	84.1	87.8	93.9	89.3	91.3	74.7	71.1	72.6
Ex. 1a	Electric bass	Mix	93.4	89.6	91.2	93.0	90.9	91.8	85.2	83.5	84.2
Ex. 1b	Piano	Mix	66.4	60.6	60.4	76.3	73.0	73.0	27.1	26.7	26.4
Ex. 1c	Guitar	Mix	69.3	56.5	59.4	73.1	62.9	65.3	38.4	33.9	35.1
Ex. 2	Electric bass	Separated	87.9	82.6	84.9	88.6	84.4	86.2	77.7	74.0	75.6
Ex. 3a	Electric bass	Mix	93.1	91.4	92.1	92.7	93.4	92.9	84.6	85.3	84.8
Ex. 3b	Piano	Mix	69.9	60.2	62.4	84.2	69.6	74.3	31.7	27.5	29.0
Ex. 3c	Guitar	Mix	76.9	54.2	60.6	76.4	59.3	64.7	45.2	36.5	39.5

The results for the experiments 0–3 can be found in Table 6.3. We see that the baseline experiment, Experiment 0, performs clearly better than the other experiments with note F_1 scores above 90 for all the instruments. For electric bass, the results are very good with note F_1 score of 99.1 and note-with-offset F_1 or 97.9 In fact, given that the electric bass test set only consists of 71 non-pitch bend tracks, many of the tracks did not have a single transcription mistake. For piano the results are also very good, beating state-of-the-art results in Table 4.1 on note F_1 score. For note-with-offset values, the piano results are around 10 percentage points behind state-of-the-art. Guitar is the most difficult instrument to transcribe with a note F_1 score of 91.3, 5.6 points behind piano and 7.8 behind electric bass bass. Transcriptions segments from the test dataset are shown in Appendix B. It is recommended to look at the frame F_1, note F_1 and note-with-offset F_1 scores in the caption to each figure to get an intuition for the metric and transcription results.

Experiment 2 has the lowest score on all the metrics of the electric bass experiments 0–3. A transcription segment for this experiment is shown in Figure 6.1. In this figure a (cropped) spectrogram is added above the transcription. We see that the model is not able to predict the first two notes but also that the audio source does not fully contain this information. At the end of the audio segment the audio source contains a distinct note, but that note is not present in the labels. This probably means that the separation model incorrectly separated a low-pitched piano or synth sound as bass.

Experiment 1 and 3 are closer in performance, but Experiment 3 is slightly better in almost all metrics.

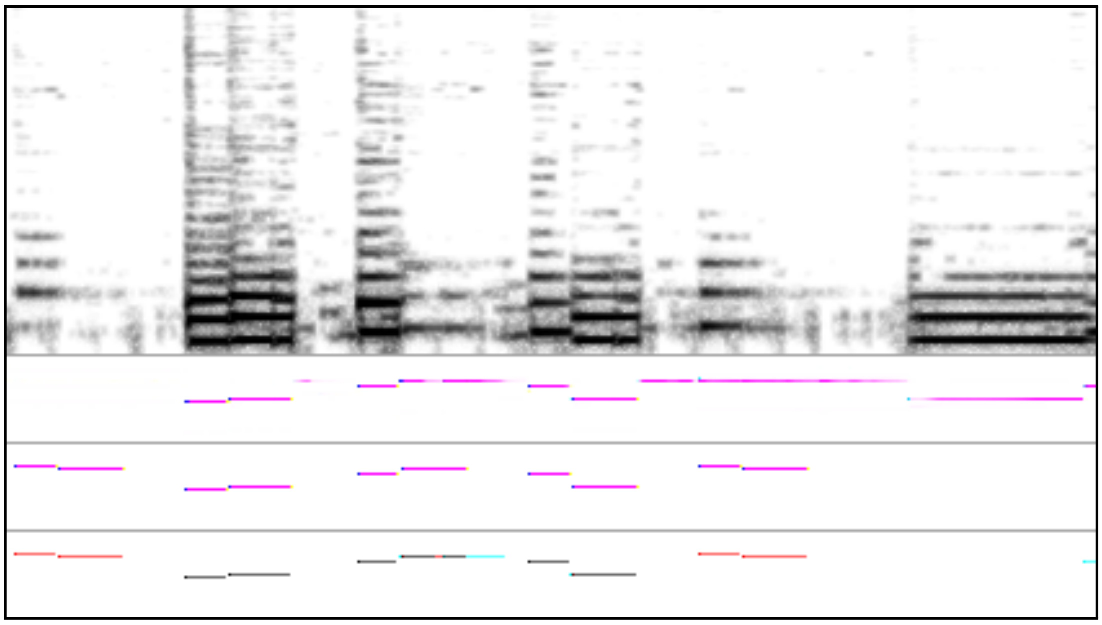

Figure 6.1.: From top to bottom: Input spectrogram, raw model predictions, reference, post-processed notes and reference together. Correct predictions are shown in black, wrong prediction in light blue and missed predictions in dark orange.

Spectrograms for the electric bass experiments 0–3 are shown in Figure 6.2. From top to bottom we see the spectrogram for the individual electric bass audio source, for the full band music ('mix'), for bass separated by Open-Unmix and finally after U-Net architecture in experiment 3. As expected, we see that the mixture spectrogram contains a lot more information than the one for individual bass. We also see that the higher frequencies for the separated bass are quite sparse and not very close to the individual audio source but that the lower, and most important, frequencies seem to be more intact. The U-Net spectrogram looks washed out compared to the one from 'mix' in most of the frequencies, but the lower frequencies are emphasized. Since the log-scaled spectrogram given as input to the U-Net architecture is symmetric around zero and that white pixel values are given to the most negative values, it explains why frequencies the U-Net model did not deem to be important are gray. If the magnitude was given as input to the U-Net and the log-scale was done afterwards, unimportant pixels would be white just as the separated spectrogram above.

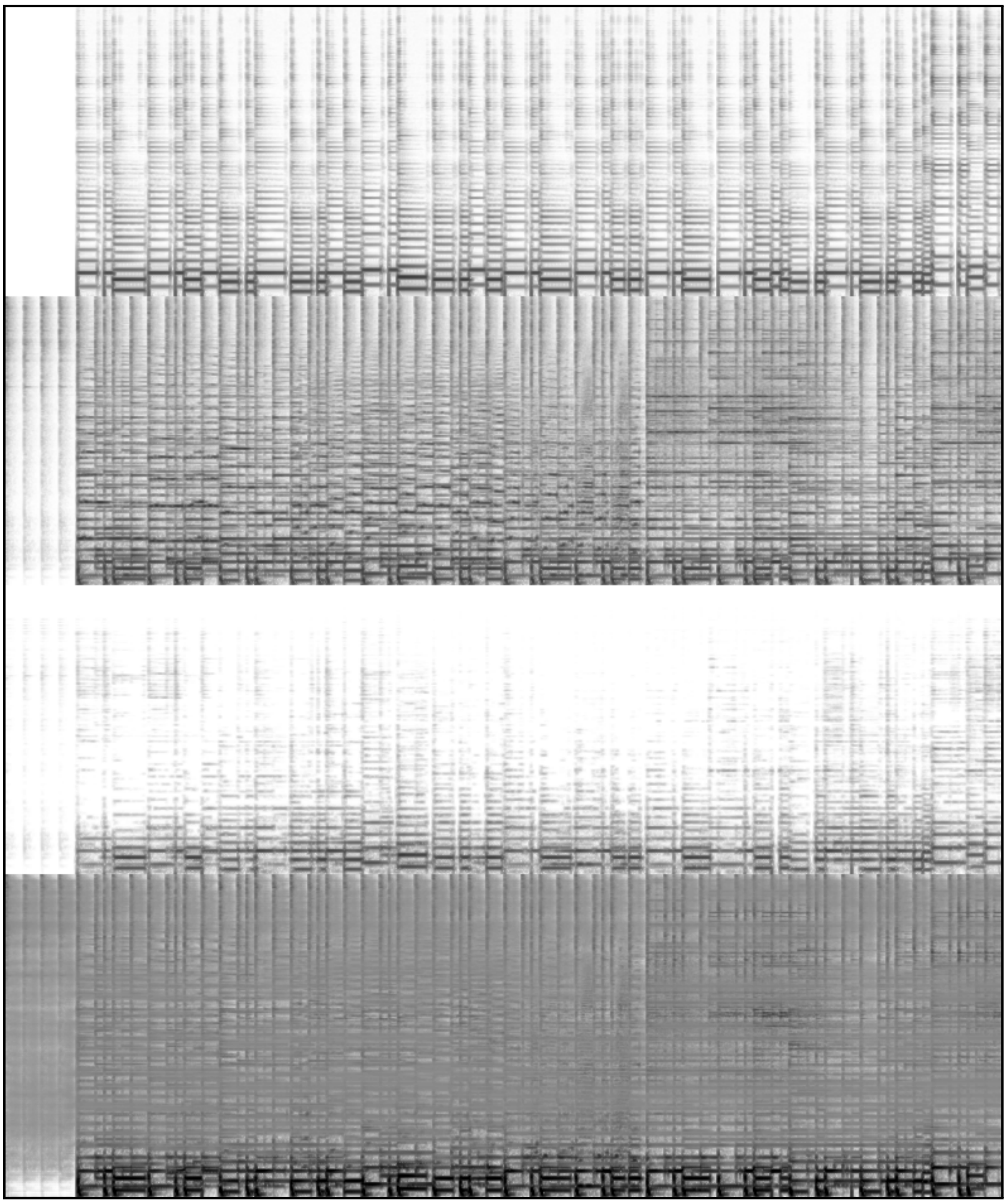

Figure 6.2.: Spectrograms for experiment 0–3. Top to bottom: individual electric bass audio, mix audio, separated audio by Open-Unmix and spectrogram after U-Net

6.3.2. Experiment 4–5

Table 6.4.: Results for experiment 4 on the modified Slakh `redux` test dataset split

	Instrument class	**Audio source**	**Frame**			**Note**			**Note w/offset**		
			P	R	F_1	P	R	F_1	P	R	F_1
Ex. 4a	All pitched	Mix	74.6	71.8	72.4	81.3	67.0	72.6	37.5	31.5	33.9
Ex. 4b	All pitched	Individual	75.5	70.9	72.3	84.1	67.4	74.0	39.8	32.5	35.4
Ex. 4c	Electric bass	Mix	-	89.6	-	-	90.3	-	-	70.6	-

The results for Experiment 4 are also shown in Table 6.4. We see that there is relatively small difference in performance between Experiment 4a and Experiment 4b even though Experiment 4a must filter out drums, sound effects and other musical noise that might be in the audio source. The model trained in these experiments are, however, only trained on a 'mix' audio source. If individual tracks were given during training larger performance difference could be seen.

Only the recall for Experiment 4c is shown since the model does not tell which instrument plays a given note and the model will predict all the other notes in the audio source. The recall gives useful information about the proportion of the electric bass notes the model is able to transcribe. We see that the recall is at the same level as Experiment 1 and 3 in Table 6.3 that operates with the same audio source on frame and note scores. For note-with-offset scores this model perform worse.

Table 6.5.: Results for experiment 5 on the Slakh `redux` test dataset split. Zero-indexed MIDI program numbers are in parenthesis after the instrument class.

Instrument class	**Tracks**	**Frame**			**Note**			**Note w/offset**		
		P	R	F_1	P	R	F_1	P	R	F_1
Piano (0–7)	143	80.7	76.9	77.4	92.1	89.6	90.6	47.6	47.1	47.2
Ch. Percussion (8–15)	23	52.4	22.4	28.6	59.0	44.1	48.2	11.7	8.4	9.4
Organ (16–23)	42	44.7	28.8	33.0	43.7	38.6	38.9	22.1	20.9	20.9
Guitar (24–31)	74	83.3	72.0	75.3	87.6	75.6	79.8	58.3	51.5	54.0
Bass (32–39)	75	96.2	94.7	95.4	95.2	96.6	95.8	89.7	90.9	90.2
Strings (40–47)	16	70.6	62.8	59.4	77.7	70.7	70.2	40.6	35.2	36.0
Ensemble (48–55)	102	83.6	49.4	59.4	77.9	51.4	58.5	46.8	31.4	35.9
Brass (56–63)	25	65.2	49.9	54.8	65.6	56.0	58.9	54.4	46.6	48.9
Reed (64–71)	26	75.5	74.4	72.0	79.4	64.3	70.0	48.5	40.4	43.4
Pipe (72–79)	13	77.7	68.7	68.1	80.3	58.2	64.3	45.4	29.0	32.9
Synth Lead (80–87)	17	44.7	32.2	36.2	43.5	43.3	42.4	27.9	27.1	26.9
Synth Pad (88-95)	39	46.1	21.2	21.8	43.2	25.3	26.1	9.8	5.2	5.6

The experimental results for experiment 5 are shown in Table 6.5 and Figure 6.3. Since an Onsets and Frames architecture is used, we would expect that the instrument classes with a clear attack would be most easily transcribed as well as instrument classes with a low level of notes played simultaneously. We see that the note-level multi-instrument model is clearly best at transcribing bass, secondly piano and guitar. The note F_1 results on there instruments are 95.8, 90.6 and 79.8. The results in this experiment is comparable

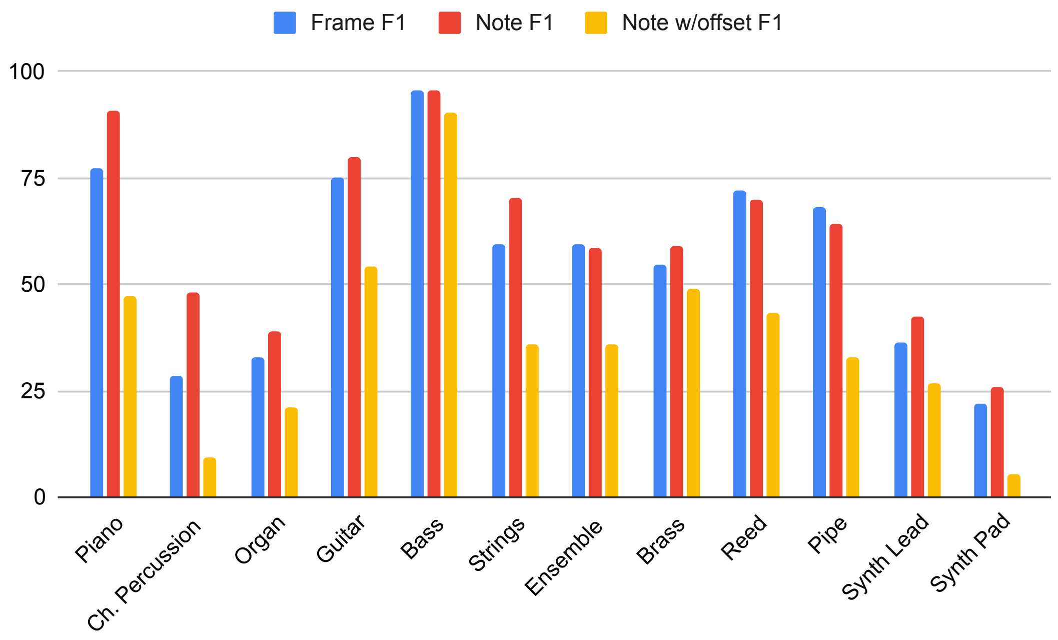

Figure 6.3.: Diagram of the results in experiment 5

to Experiment 0 since it also transcribes individual audio source (but acoustic bass and synth bass added to the bass instrument class in this experiments). The bass performance in this experiment is only slightly worse than the baseline even though the model did not use any individual bass audio during training. A larger drop relative drop in performance is seen for piano and even larger for guitar.

We see that the model struggles the most with synth pad. This is not surprising since this instrument class does not have a clear attack. It is more surprising that chromatic percussion, organ and synth lead are among the most difficult to transcribe. An evaluation of this, as well as a figure that show the performance where octave errors are neglected are in Section 7.1.1 and shown in Figure 7.2 (page 52).

6.3.3. Experiment 6

Table 6.6.: Results for experiment 6

Instrument class	**Tracks**	**Frame**			**Note**			**Note w/offset**		
		P	R	F_1	P	R	F_1	P	R	F_1
Piano (0–7)	27	65.2	63.1	63.0	66.6	71.4	67.3	25.5	28.8	26.5
Guitar (24–31)	27	68.3	45.9	51.2	68.6	50.8	55.6	29.6	22.6	24.7
Bass (32–39)	27	92.7	87.2	89.5	91.1	89.1	89.9	79.3	77.6	78.3
All pitched (0–95)	27	82.2	75.3	78.3	81.2	70.2	74.4	44.2	38.4	40.6
Drum	27	68.6	64.2	65.2	75.5	66.9	69.9	75.5	66.9	69.9

We see the results for experiment 6 in Table 6.6 and Figure 6.4. This stream-level model is able to transcribe piano, guitar, bass, all pitched instrument and drum simultaneously from a mixture audio source. Again we see that bass is most easily transcribed. The multi-instrument audio source is comparable to Experiment 1–3 and we see that the bass performance is almost as good as Experiment 1 which uses the same architecture. The results for the all pitched category are slightly better than Experiment 4a.

The drum score are calculated in the same manner as the pitched instruments. This explains why the note and note-with-offset score are equal.

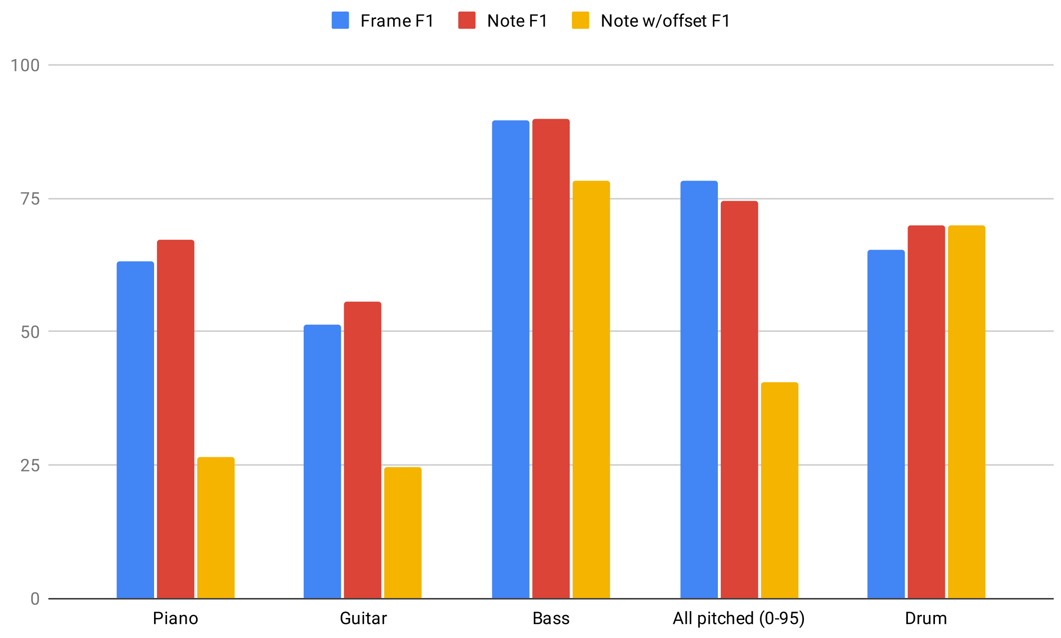

Figure 6.4.: Diagram of the results in experiment 6

7. Evaluation and Conclusion

The goal of this Master's Thesis was to extend automatic music transcription (AMT) to a multi-instrument setting. Several experiments were conducted to achieve this goal. The first set of experiments investigate different architectures and if music source separation pre-processing improves performance. These experiments show that the current single-instrument AMT model works well on a mixture audio source, and can be further enhanced by using a joint music source separation and AMT architecture. Music source separation pre-processing did not improve performance, but the model was not fine-tuned on the used dataset. The next experiment shows that it is possible to train a universal note-level AMT model solely on mixture audio. This model reaches a note F_1 score of **90.6** on piano and **95.8** on bass, only slightly behind the current state-of-the-art. The transcription performance varies greatly between instrument classes and the note-with-offset scores are still far behind the current single-instrument for all instrument classes except bass. Finally, a stream-level model is trained that can transcribe piano, guitar, bass, drum and all the pitched instruments simultaneously in 5-10 times real-time performance on CPU and 10-100 times real-time performance on GPU. All the experiments are conducted on the synthetically rendered MIDI dataset Slakh (see Section 3.7). During the work on this dataset, several systematic and non-systematic errors were found and reported to the creators of the dataset.

The following chapter will evaluate the findings in light of the initial Goal and Research Questions. In addition, a discussion section is dedicated to the implications of the results, as well as addressing additional questions which might have arisen based on the findings. Finally, the contributions of this Master's Thesis are summarized, and the Thesis is ended with suggestions for future work.

7.1. Evaluation

Looking back to the introduction, this Master's Thesis was written with a single goal in mind: Introduce multi-instrument automatic music transcription models. This section will first evaluate the initial Research Questions, and finally evaluate whether the answers to the Research Questions have contributed to reaching the overall Goal.

7.1.1. Evaluation of Research Questions

Experiment 0 served as an upper baseline for all the Research Questions. The results of Experiment 0a show that automatic music transcription more or less is a solved problem for electric bass with a single-instrument audio source. This experiment achieved a frame F_1 score of **98.5**, note F_1 score of **99.1** and note-with-offset F_1 score of **97.9** on the modified Slakh test dataset. In fact, the automatic music transcription model could almost be used as an efficient lossless compression for this instrument just at speech-to-text can for voice. To reach this score on electric bass some systematic errors in the Slakh dataset had to be addressed as well as some errors for individual stems. One systematic error was that all the notes in the highest octave in the labels were not present in the audio. This error occurred in the generation of the dataset since the notes were truncated one octave higher than the range of a 5-string electric bass. That also means that the seven deepest notes on a 4-string bass are not present in the dataset, which is unfortunate since these are very common notes on electric bass. The maximum simultaneous harmony for electric bass was also restricted to two notes at the same time to remove wrongly specified stems in the dataset (some stems looked like a keyboard classified as electric bass with only playing chords). Several stems were also found to have errors and were removed from the dataset during training and evaluation. A comprehensive list of the tracks removed can be found in Appendix A.

The piano results in Experiment 0b show that the baseline reached a better note F_1 score, **96.9**, than the best state-of-the-art results in Table 4.1 (page 25). This happens even though Experiment 0b is evaluated on the entire piano instrument class (MIDI programs 0–7) while the related work is only evaluated on acoustic pianos. Since the same architecture as in the related work is used, this probably reflects that the piano instrument class in Slakh is an easier dataset than MAESTRO (see Section 3.2). MAESTRO consists of classical performances by virtuous pianist which probably means that the dataset have more fast notes and an higher level of notes played simultaneously. For note-with-offset results, Experiment 0b is worse than the results in the related work. This could be explained by inaccuracies in the labels and that the model in this experiment is only trained around one-tenth the number of epochs in the related work. By observing the validation results during training in Figure 7.1 we see in the note-with-offsets graph to the right seem to have a slower upward-trend, at least for guitar and piano. This can indicate that this metric would benefit the most for longer training.

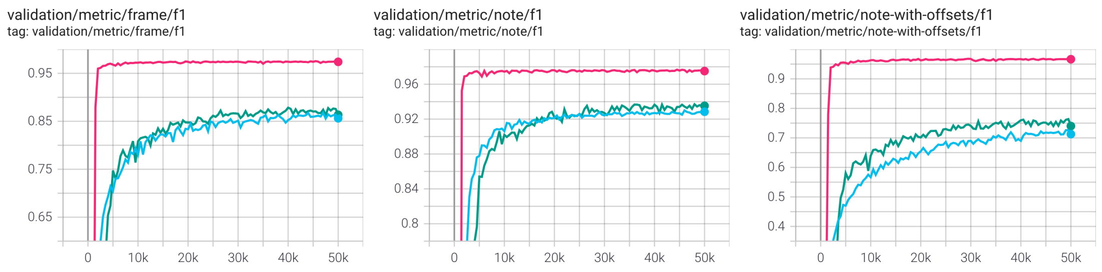

Figure 7.1.: Validation F_1 scores during training for Experiment 0. Purple is electric bass, cyan piano and green guitar.

Experiment 0c shows that guitar is the most difficult instrument to transcribe at least when it comes to frame and note results. For note-with-offset scores, guitar is slightly better than piano.

Research question 1 *How well do state-of-the-art single-instrument models perform in a multi-instrument setting without music source separation?*

Experiment 1 was directly designed to answer Research Question 1. This experiment shows that it indeed is possible to use the Extended Onsets and Frames model in a multi-instrument setting. For electric bass, the note F_1 score of **91.8** in the multi-instrument setting is comparable to the state-of-the-art single-instrument piano results. Experiment 1b and 1c also show that it is possible to transcribe piano and guitar in this setting with note F_1 scores of **73.0** and **63.3**, respectively. These results are better than the single-instrument state-of-the-art piano results from 2016 (Sigtia et al., 2016; Kelz et al., 2016) (see Table 4.1, page 25). However, as expected, there is a significant performance reduction to the baseline with a relative note F_1 drop of 7.4%, 25% and 28% for electric bass, piano and guitar, respectively.

A much larger relative reduction was seen for note-with-offset F_1 scores with 14%, 63% and 51% for the same instruments. Note-with-offset clearly is the harder metric and this is further emphasized when the audio source is more difficult. Given the post-processing steps (see Section 5.3) this is not too surprising. If only a single frame prediction is below the chosen frame threshold the note will be too short. This is in contrast to the onset prediction where each onset above the onset threshold will be the start of a note. A further study of the note-with-offset mistakes would be valuable. If the majority of the note-with-offset mistakes were due to too short notes this could be addressed in the post-processing step by for instance averaging the consecutive frame predictions at the same pitch.

Research question 2 *How well do state-of-the-art single-instrument models perform in a multi-instrument setting when the audio is source separated by a pre-trained music source separation model?*

Experiment 2 was directly designed to answer Research Question 2. Since the pre-trained music source separation model only separates into vocals, drums, bass and accompaniment, and there are no vocals in the Slakh dataset, only an experiment for bass was conducted. The first attempt at using a pre-trained model, Spleeter by Hennequin et al. (2020), was unsuccessful since the model was not able to separate the bass at all. Subsequently, the Open-Unmix model by Stöter et al. (2019b) was used with better separating results. As can be seen in Table 6.3, Experiment 2 got the worst score in all of the electric bass experiments with a note F_1 score of **86.2** and a note-with-offset score of **75.6**. This can to a large degree be explained by the spectrograms shown in Figure 6.1 and Figure 6.2. The first figure shows that the spectrogram is not following the labels and the second picture shows that the separated spectrogram and individual bass spectrogram look quite different.

One aspect of the experiments that could be changed if redone, is the choice of only using electric bass and not also including the synth bass stems from the dataset. Electric bass was chosen since it was used by Hung et al. (2020), but after listening to an excerpt from the MUSDB18 dataset Open-Unmix is trained on, is it clear that several tracks use synth bass in the bass stems. This can also explain the mediocre results since synth bass can be added to the separated audio. In addition, fine-tuning the music source separation model on the Slakh test set would probably increase separation quality considerably given than Spleeter works well on real audio but did not work at all on the Slakh dataset.

Research question 3 *How well does a new architecture that joins source separation with automatic music transcription perform in a multi-instrument setting?*

Experiment 3 was directly designed to answer Research Question 3. This experiment has a better F_1 score than Experiment 1 in all the metrics except note scores for guitar where it is marginally worse. For the other F_1 score the improvement in performance is not drastic, only a few relative percent improvements. Nevertheless, the results show that the addition of the U-Net model overall improves performance. Figure 6.2 (page 43) also shows that the U-Net model emphasizes logical frequencies for bass.

Table 6.2 (page 40) shows that the model size is increased from 21.6MB to 32.8MB, over a 50% relative increment, to the baseline model. Since the increase in model performance is so modest, this might not be a favorable trade-off. The training and inference time was, however, not affected by the U-Net architecture to a large degree.

In Experiment 3 only the automatic music transcription labels were used. Further investigation on using the separated audio as labels would be interesting. Would the performance be any better and if, by how much? Loading more audio during training would increase training time considerably since this is one of the most time-consuming parts or require much larger RAM if the audio is kept in memory.

Furthermore, the U-Net architecture could probably be used more efficiently as a backbone for automatic music transcription. In the Extended Onsets and Frames with U-Net architecture (see Section 5.2, page 31), the U-Net is only used as additional computing power to modify the input spectrogram. But since this architecture can separate an audio source into several stems in a music source separation setting, it needs to have higher-level information such as when an instrument class is active and the note onsets in addition to the specific frequencies the separated stems consist of. The layers before the last layer in the U-Net are bigger in terms of the number of parameters and should contain this high-level information. It would be interesting to use the second-to-last layer in the U-Net model or several layers from the decoder to the transcription head. Since the U-Net architecture works so well for music source separation, and music source separation seems to be an even harder problem than automatic music transcription, maybe the U-Net model is sufficient for good transcription results?

Research question 4 *How well does a note-level multi-instrument automatic music transcription perform in a single-instrument setting?*

Experiment 5 was directly designed to answer Research Question 4. The results of this experiment show that the single-instrument transcription performance is highly dependent on the instrument class, at least in this dataset. However, for the three instrument classes used in the baseline, the note F_1 drop are not very high. The note F_1 scores for bass, piano and guitar are **90.2** (3.1% drop), **90.6** (6.5% drop) and **79.8** (13% drop), respectively. Again we see that the note-with-offset is a more difficult metric and the relative drop to the baseline is higher (7.8%, 34% and 26% for the same instruments). The model trained for this experiment only used the mixture audio as training data. This means that the results for Experiment 5 are quite far from the audio source the model was trained on unless some songs had longer parts with only a single instrument class.

The three instruments from the baseline are, however, the three instruments with the best scores in Experiment 5. For synth pad, the note F_1 is as low as 26.1 and the other instruments are at least ten percentage points behind guitar. A lower score for synth pad is expected since it has a slow non-distinct attack that does not fit the Onsets and Frames family of architectures well. Many of the other instruments have gotten an unreasonable low score either due to systematic octave misclassifications when rendering the audio or that the strength of the partials are so far away from the most common notes in the dataset so that octave errors occur. Figure 7.2 shows the frame F_1 results with the chroma results in a darker blue color. The chroma results are calculated in the same manner as the frame results except that the notes are truncated to the same octave. This figure shows that the note-level model has many octave errors, especially for organ, brass and synth lead. With the chroma results, piano, guitar, brass, reed and pipe are almost at the same level. For many applications, octave errors are not severe, at least when they are consistent (it is for instance very easy to change octave in most note editing software).

Table 6.5 shows the number of tracks for each instrument class in the test set in addition to the metrics. We see that the number of tracks varies greatly. Piano have 143 tracks while pipe, strings and synth lead have 13, 16 and 17 tracks, respectively. This tells that some of the results should be given more confidence.

Research question 5 *How well does a stream-level multi-instrument automatic music transcription perform?*

Experiment 6 was directly designed to answer Research Question 5. The results show that it is possible to train a stream-level automatic music transcription model with good results. The results from this experiment are directly comparable to Experiment 1 which operates on the same instruments and labels and uses the same architecture. Compared to Experiment 1 the results are only slightly worse with note F_1 scores of **73.0** for piano (7.8% drop), **65.3** for guitar (15% drop) and **91.8** for bass (2.1%). In comparison, the piano results are still better than state-of-the-art single-instrument piano results from 2016 (Sigtia et al., 2016; Kelz et al., 2016) and around 20% relative drop from 2018 (Hawthorne et al., 2018). For the all-pitched instrument class, the performance is better than the note-level results from Experiment 4a with a note F_1 score of **74.4** and note-with-offset F_1 score of **40.6**. This is a 2.5% and 20% relative improvement to Experiment 4a. It is reasonable that adding additional labels improves this transcription category. If the model knows that a note is played by a given instrument it might give

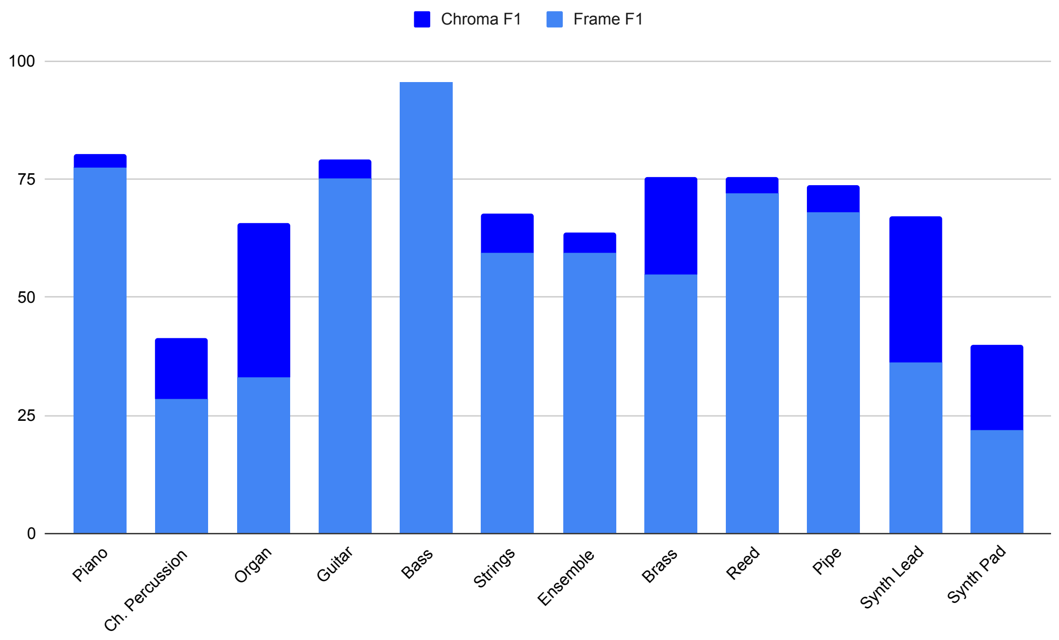

Figure 7.2.: Frame F_1 results in experiment 5 with chroma

more confidence in the prediction. The reason why the model is worse for bass, piano and guitar might only be because the bigger models need longer training to achieve the same score (both experiments were trained 50 000 iterations).

Drums were added in this experiment to have a complete rock band automatic music transcription model. For the sake of convenience and computing efficiency, this model is very appealing. The model is almost twice as big as the baseline Extended Onsets and Frames models since the number of parameters in the last linear layers are increased for the larger concatenated labels. However, the training and inference time are not influenced much by the larger size. For deploying the model in production it certainly is much easier to have one larger model than five different models. The five different models would be much larger and have longer inference time.

7.1.2. Evaluation of the Main Goal

With background in the results obtained when answering the five Research Questions, an introduction of several multi-instrument automatic music transcription models has been made, which completes the main Goal of this Master's Thesis. The first three Research Questions were concerned with different model architectures and music source separation pre-processing. The result from this study shows that the state-of-the-art single-instrument automatic music transcription model architecture works well in a multi-instrument setting. In fact, the model archives better piano note F_1 scores than the single-instrument state-of-the-art approaches from 2016 (Sigtia et al., 2016; Kelz et al., 2016) without any modifications. For electric bass, the multi-instrument score is even better, competing with state-of-the-art single-instrument piano scores from 2018 (Hawthorne et al., 2018). The experiments in this Thesis, however, are done on synthetically rendered audio while the reference results are on real piano records so they are not fully comparable. An increase in performance was seen with a joint music source separation and automatic music transcription model, but the increment was modest and with nearly twice as many model parameters this might not be a favorable trade-off.

The last two Research Questions were concerned with note-level and stream-level models. The results show that it is possible to train both kinds of models with decent results. A note-level model trained on a full band audio source is able to transcribe piano with a note F_1 score of **90.6**, only a few percentage points behind the current state-of-the-art, albeit on a presumably easier dataset. A stream-level automatic music transcription model is now also possible for piano, guitar, bass, drum and all pitched instruments with above 50 note F_1 scores for all instrument classes. For note-with-offsets scores, the multi-instrument models still are far behind the single-instrument state-of-the-art results in all the experiments.

7.2. Discussion

The following section provides a discussion of the most influential part of this work. First, a discussion of the limitations and pitfalls of the dataset is presented, as well as the related work that has used this dataset. Secondly, a discussion of the assumptions and possible modifications to the Onsets and Frames architecture is presented.

7.2.1. Dataset

A machine learning task cannot be better than the data the models are trained and evaluated on. Therefore a discussion on the limitations and pitfalls on the dataset used in this work is needed. Several stems contain some kind of errors, for a full list see Appendix A. These errors range from stems where the audio only consist of white noise, corrupted MIDI notes, the audio and labels are at different pitches, and the length of the labels and audio are not matched. The list in Appendix A certainly is not exhaustive and mostly consists of electric bass stems. By removing erroneous stems, addressing systematic errors in the bass octave labels, and removing pitch-bend tracks the results in Experiment 1a and the corresponding result in Grønbech (2020) went from a note F_1 score of **65.2** to **91.8**, a 41% relative improvement. For Experiment 4a the change is less dramatic, but still present at a note F_1 score of **63.5** to **72.6**. How much the performance improvement stems from the modifications to the training set or test set is not clear, but it nevertheless shows the importance of the dataset.

Only Hung et al. (2020) in the related work have used the Slakh dataset for automatic music transcription. They have reported the score in the less commonly used note accuracy metric. It is not clear from their paper if they calculated the frame accuracy from `mir_eval` library or if it is based on the note onsets. Since the code is not open source it is also not possible to check the calculation. In addition, Hung et al. (2020) based the results on the `splits_v2` split which has duplicate tracks and only the instruments electric bass, acoustic piano and distorted guitar was considered. Nevertheless, if the frame accuracy was meant, Experiment 3 has almost comparable results to Hung et al. (2020) with frame accuracy scores of 87.5% to 86.1% for electric bass, 47.2% to 51% for piano and 46.9% to 61% for guitar (best mixture results from Hung et al. (2020) is chosen). Given the uncertainty in the calculation of the metrics and since Hung et al. have not reported the MIDI programs they have chosen, a fair comparison is not possible.

For reproducibility on the Slakh dataset a PyTorch data-loader[1] has been created during the work on this Thesis and is readily available on PyPI[2]. This data-loader addresses some systematic errors in the Slakh dataset such as removing the labels from the higher bass frequency since the for these notes are not present. In addition, is filters out all the tracks with known errors from Appendix A. Ideally, a regeneration of the Slakh dataset should be made which removes the duplicated tracks and addresses the tracks with errors and systematic errors in the dataset.

The most important contribution for further progress on multi-instrument automatic music transcription is probably a new high quality dataset. Just as MAPS (see Section 3.1) has a synthesized training set and a real-audio test set, a real-audio test set for multi-instrument music transcription would benefit the research community greatly. As mentioned in the introduction, human-labeling such a dataset would be an incredible time-consuming task, but using the note-level model from Experiments 4–5 as a pre-process could save a lot of time. Also, if the music were originally played by sheet music, an alignment could be done. A real-audio test set would show the generalizability of the automatic music transcription models.

7.2.2. Model Architecture

The model architecture used in this work is based on the Onsets and Frames architecture that has been used in the related work for piano transcriptions. One assumption in this architecture is that the note onset is the most perceptual distinct part of the note. This is also reflected in the post-processing step where the note onsets are given special consideration since only the onset predictions can be the start of the note. One aspect that might seem odd when looking at the architecture in Figure 4.1 (page 23) is that only about one-fifth of the computing power goes to the onsets. No information from the other detection head goes to the onset. But since the onsets are so distinct for piano more computing power is not needed for good results. For many other instruments, however, the onset is not the most perceptual part, such as for synth pads and can be for wind instruments. Therefore, a modification to the architecture that gives more computing power to the onsets and information flow from the frame detector to the onsets detector can be investigated. One straight-forward modification would be to change the frame

[1]https://github.com/greenbech/slakh-pytorch-dataset
[2]https://pypi.org/project/slakh-dataset

and onset label in Figure 4.1. This would give more information to the most import predictions, the onset predictions, but at the cost of lower less frame computing power. As shown in Experiment 6, concatenating several labels at the detection head does not seem to degrade the performance. Maybe a simpler architecture where onsets and frame were predicted as the same detector head would give better results?

7.3. Contributions

The contributions of this Master's Thesis are manifold. This is the first work that investigates different architectures and music source separation pre-processing for multi-instruments automatic music transcription. The results of this work show that it is possible to transcribe one instrument from a mixture source using the current state-of-the-art single-instrument architecture. For piano and guitar the score in this setting is better than the best single-instrument piano scores from 2016 (Sigtia et al., 2016; Kelz et al., 2016). For electric bass the score is even better at an note F_1 score of **91.8**, almost as good at the current single-instrument state-of-the-art scores.

The experiments on note-level automatic music transcription show that it is possible to train a multi-instrument model on a mixture source and reach good performance on isolated tracks. This model shows that it is possible to train one model to transcribe many different instruments which can be valuable in many settings. If the model was fine-tuned on isolated tracks the performance could have been even better.

Furthermore, the results in this Thesis shows that a stream-level automatic music transcription is possible. This model can transcribe a full rock band simultaneously.

Finally, a PyTorch data-loader for Slakh has been created. This data-loader removes systematic and individual errors from the Slakh dataset. With this data-loader it is possible to load the audio and several instrument classes in a efficient and convenient way. During the work in this Thesis is was found that the bass stems in Slakh did not contain the deepest octave due to an error in the generation of the dataset. Likewise, the audio for the highest octave in bass are not present. These labels are removed when using the data-loder. The data-loader enables reproducible results for researchers in the future.

7.4. Future Work

As mentioned in the discussion, the dataset is at the heart of every machine learning task. During the work of this Thesis, several errors in the Slakh dataset (see Section 3.7) were found. For further work improvement and fair comparisons in this field one of the first things to look into should be to obtain or create a better dataset. This could either be to re-render and fix the errors in the Slakh dataset, combine the existing dataset or create a new one. For a multi-instrument model, the MAESTRO dataset (see Section 3.2), the Expanded Groove MIDI Dataset (see Section 3.3) and MusicNet (see Section 3.4) could be combined in training for more variation. MusicNet could serve as a real-audio test set, but since a relatively high error rate is reported this would need further investigation. In addition, a real-audio annotated dataset with vocals would be highly beneficial for transcribing a typical pop or rock band. Maybe the existing music source separation dataset MUSDB18 (see Section 3.8) could be transcribed for an automatic music transcription use-case? Given the performance of the note-level multi-instrument automatic music transcription model from Experiment 5, this model could be used as a pre-process before being corrected by humans.

A limitation of the work in this project is that tracks with pitch bends are disregarded. An investigation of parsing MIDI files with pitch bend to a piano-roll format as well as supporting pitch bends in the model architecture would be a natural further extension of the work. One strategy for incorporating pitch bends in the current architecture would be to add an additional detector head that would predict the relative pitch bend for a given note. Another idea could be to linearly interpolate between the frame prediction.

Experiment 3 shows that adding a music source separation model as a backbone to the automatic music transcription model improves performance slightly. It would be interesting to use the separated audio as labels in training for the U-Net and see if this improves performance. By doing this a combined music source separation and automatic music transcription model could be made.

Additionally, experiments on using the U-Net architecture directly for automatic could be an interesting direction. Since the U-Net model is able to separate music, it needs to have much the same high-level information as is needed in multi-instrument automatic music transcription such as when an instrument is playing and when the onset is. Since the U-Net model does not have an recurrent neural network this could be a highly efficient automatic transcription model.

In addition, new deep learning architectures from other fields could be tested in an music setting to improve performance. The recent architecture Transformer from natural language processing could be used instead of BiLSTM. But due to the quadratic dependence on the input size, reaching a reasonable performance on an audio source could be challenging. More recently, the FNet architecture seems to have comparable performance to the transformer while being more efficient and only scales $\mathcal{O}(N \log N)$ with the input size (Lee-Thorp et al., 2021). The new deep learning operation Involution could also substitute some or all the convolution layers in the current architectures (Li et al., 2021).

Experiment 5 shows that a universal note-level multi-instrument model is possible, but the performance varies between different instrument classes. An investigation of fine-tuning such a model would be interesting. How many notes from a new instrument are needed before the performance increases significantly?

Furthermore, Experiment 6 shows that it is possible to train a stream-level multi-instrument model and that concatenating additional instrument classes in the label does not seem to affect the performance negatively greatly. An investigation on solving more Music Information Retrieval tasks from the same model would be interesting. Would the same model be able to predict beats and down-beats in addition to the notes? What about music source separation or predicting the mood of the music? It might be that the internal model representation would be better if several tasks were solved simultaneously.

Finally, creating a new post-processing step for the current multi-instrument model to extract the chords should be feasible with the current performance. In many real-world applications having the chords are more valuable than having a transcription. Pianists and guitar players often play after chords, not sheet music. When transcribing the bass and having a high frame performance, chord prediction should be possible. This would presumable be more fitted for a rule-based approach due to the lack of training data.

Bibliography

Mert Bay, Andreas F. Ehmann, and J. Stephen Downie. Evaluation of Multiple-F0 Estimation and Tracking Systems. In *Proceedings of the 10th International Society for Music Information Retrieval Conference*, pages 315–320, Kobe, Japan, October 2009. ISMIR. doi:10.5281/zenodo.1418241. 13

Emmanouil Benetos and Simon Dixon. Joint multi-pitch detection using harmonic envelope estimation for polyphonic music transcription. *Selected Topics in Signal Processing, IEEE Journal of*, 5:1111–1123, November 2011. doi:10.1109/JSTSP.2011.2162394. 19, 20

Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. Automatic music transcription: Challenges and future directions. *Journal of Intelligent Information Systems*, 41:407–434, December 2013. doi:10.1007/s10844-013-0258-3. 19

Emmanouil Benetos, Simon Dixon, Zhiyao Duan, and Sebastian Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36:20–30, January 2019. doi:10.1109/MSP.2018.2869928. 8, 9

Thierry Bertin-Mahieux, Daniel P. W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Society for Music Information Retrieval Conference*, pages 591–596. University of Miami, 2011. 17

Rachel M. Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. MedleyDB: A Multitrack Dataset for Annotation- Intensive MIR Research. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 155–160. ISMIR, October 2014. doi:10.5281/zenodo.1417889. 18

Sebastian Böck and Markus Schedl. Polyphonic piano note transcription with recurrent neural networks. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–124, 2012. doi:10.1109/ICASSP.2012.6287832. 21

Lee Callender, Curtis Hawthorne, and Jesse H. Engel. Improving perceptual quality of drum transcription with the expanded groove MIDI dataset. *CoRR*, abs/2004.00188, 2020. URL `https://arxiv.org/abs/2004.00188`. 16

Valentin Emiya, Roland Badeau, and Bertrand David. Multipitch estimation of piano sounds using a new probabilistic spectral smoothness principle. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6):1643–1654, 2010. doi:10.1109/TASL.2009.2038819. 15, 20

Jon Gillick, Adam Roberts, Jesse H. Engel, Douglas Eck, and David Bamman. Learning to groove with inverse sequence transformations. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 2269–2279. PMLR, 2019. URL `http://proceedings.mlr.press/v97/gillick19a.html`. 16

Henrik Grønbech. Automatic music transcription with deep learning. Specialization Project, Dept. of Computer Science, Norwegian University of Science and Technology, Trondheim, Norway, December 2020. iii, 7, 15, 19, 29, 30, 53

Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and Frames: Dual-Objective Piano Transcription. In *Proceedings of the 19th International Society for Music Information Retrieval Conference*, pages 50–57, Paris, France, September 2018. ISMIR. doi:10.5281/zenodo.1492341. iii, 13, 16, 22, 23, 24, 25, 29, 51, 53

Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019. URL `https://openreview.net/forum?id=r1lYRjC9F7`. 16, 23, 24, 25, 29

Romain Hennequin, Anis Khlif, Felix Voituret, and Manuel Moussallam. Spleeter: a fast and efficient music source separation tool with pre-trained models. *Journal of Open Source Software*, 5(50):2154, 2020. doi:10.21105/joss.02154. URL `https://doi.org/10.21105/joss.02154`. Deezer Research. 26, 32, 36, 49

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735. 11

Yun-Ning Hung, Gordon Wichern, and Jonathan Le Roux. Transcription is all you need: Learning to separate musical mixtures with score as supervision. *CoRR*, abs/2010.11904, 2020. URL `https://arxiv.org/abs/2010.11904`. 24, 36, 50, 53, 54

Andreas Jansson, Eric J. Humphrey, Nicola Montecchio, Rachel M. Bittner, Aparna Kumar, and Tillman Weyde. Singing voice separation with deep u-net convolutional networks. In Sally Jo Cunningham, Zhiyao Duan, Xiao Hu, and Douglas Turnbull, editors, *Proceedings of the 18th International Society for Music Information Retrieval Conference, ISMIR 2017, Suzhou, China, October 23-27, 2017*, pages 745–751, 2017. URL `https://ismir2017.smcnus.org/wp-content/uploads/2017/10/171_Paper.pdf`. iii, 31

Rainer Kelz, Matthias Dorfer, Filip Korzeniowski, Sebastian Böck, Andreas Arzt, and Gerhard Widmer. On the potential of simple framewise approaches to piano transcription. In Michael I. Mandel, Johanna Devaney, Douglas Turnbull, and George Tzanetakis, editors, *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016, New York City, United States, August 7-11, 2016*, pages 475–481, 2016. URL `https://wp.nyu.edu/ismir2016/wp-content/uploads/sites/2294/2016/07/179_Paper.pdf`. 22, 24, 25, 29, 49, 51, 53, 55

Rainer Kelz, Sebastian Böck, and Gerhard Widmer. Deep polyphonic ADSR piano note transcription. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2019, Brighton, United Kingdom, May 12-17, 2019*, pages 246–250. IEEE, 2019. doi:10.1109/ICASSP.2019.8683582. URL `https://doi.org/10.1109/ICASSP.2019.8683582`. 23

Jong Wook Kim. Onsets and frames. `https://github.com/jongwook/onsets-and-frames`, 2019. 38

Jong Wook Kim and Juan Pablo Bello. Adversarial learning for improved onsets and frames music transcription. In Arthur Flexer, Geoffroy Peeters, Julián Urbano, and Anja Volk, editors, *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019, Delft, The Netherlands, November 4-8, 2019*, pages 670–677, 2019. URL `http://archives.ismir.net/ismir2019/paper/000081.pdf`. iii, ix, 23, 24, 25

Qiuqiang Kong, Bochen Li, Xuchen Song, Yuan Wan, and Yuxuan Wang. High-resolution piano transcription with pedals by regressing onsets and offsets times. *CoRR*, abs/2010.01815, 2020. URL `https://arxiv.org/abs/2010.01815`. 22, 24, 25

James Lee-Thorp, Joshua Ainslie, Ilya Eckstein, and Santiago Ontañón. Fnet: Mixing tokens with fourier transforms. *CoRR*, abs/2105.03824, 2021. URL `https://arxiv.org/abs/2105.03824`. 56

Duo Li, Jie Hu, Changhu Wang, Xiangtai Li, Qi She, Lei Zhu, Tong Zhang, and Qifeng Chen. Involution: Inverting the inherence of convolution for visual recognition. *CoRR*, abs/2103.06255, 2021. URL `https://arxiv.org/abs/2103.06255`. 56

Antoine Liutkus, Fabian-Robert Stöter, Zafar Rafii, Daichi Kitamura, Bertrand Rivet, Nobutaka Ito, Nobutaka Ono, and Julie Fontecave. The 2016 signal separation evaluation campaign. In *Latent Variable Analysis and Signal Separation*, volume 10169 of *Lecture Notes in Computer Science*, pages 323–332, February 2017. ISBN 978-3-319-53546-3. doi:10.1007/978-3-319-53547-0_31. 18

Ethan Manilow, Gordon Wichern, Prem Seetharaman, and Jonathan Le Roux. Cutting music source separation some slakh: A dataset to study the impact of training data quality and quantity. In *2019 IEEE Workshop on Applications of Signal Processing* *to Audio and Acoustics, WASPAA 2019, New Paltz, NY, USA, October 20-23, 2019*, pages 45–49. IEEE, 2019. doi:10.1109/WASPAA.2019.8937170. iii, 3, 17, 18, 38, 39

Matija Marolt. A connectionist approach to automatic transcription of polyphonic piano music. *Multimedia, IEEE Transactions on*, 6:439–449, July 2004. doi:10.1109/TMM.2004.827507. 21

Nobutaka Ono, Zafar Rafii, Daichi Kitamura, Nobutaka Ito, and Antoine Liutkus. The 2015 signal separation evaluation campaign. In Emmanuel Vincent, Arie Yeredor, Zbynek Koldovský, and Petr Tichavský, editors, *Latent Variable Analysis and Signal Separation - 12th International Conference, LVA/ICA 2015, Liberec, Czech Republic, August 25-28, 2015, Proceedings*, volume 9237 of *Lecture Notes in Computer Science*, pages 387–395. Springer, 2015. doi:10.1007/978-3-319-22482-4_45. 18

Colin Raffel. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching.* PhD thesis, Department of Electrical Engineering, Columbia University, New York, New York, July 2016. iii, 17, 38

Colin Raffel, Brian McFee, Eric J. Humphrey, Justin Salamon, Oriol Nieto, Dawen Liang, and Daniel P. W. Ellis. MIR_EVAL: A Transparent Implementation of Common MIR Metrics. In *Proceedings of the 15th International Society for Music Information Retrieval Conference*, pages 367–372, Taipei, Taiwan, October 2014. ISMIR. doi:10.5281/zenodo.1416528. 3, 41

Zafar Rafii, Antoine Liutkus, Fabian-Robert Stöter, Stylianos Ioannis Mimilakis, and Rachel Bittner. The MUSDB18 corpus for music separation, December 2017. URL `https://doi.org/10.5281/zenodo.1117372`. 18

Alexander Schindler, Rudolf Mayer, and Andreas Rauber. Facilitating Comprehensive Benchmarking Experiments on the Million Song Dataset. In *Proceedings of the 13th International Society for Music Information Retrieval Conference*, pages 469–474, Porto, Portugal, October 2012. ISMIR. doi:10.5281/zenodo.1417521. 17

Prem Seetharaman, Gordon Wichern, Bryan Pardo, and Jonathan Le Roux. Autoclip: Adaptive gradient clipping for source separation networks. In *30th IEEE International Workshop on Machine Learning for Signal Processing, MLSP 2020, Espoo, Finland, September 21-24, 2020*, pages 1–6. IEEE, 2020. doi:10.1109/MLSP49062.2020.9231926. URL `https://doi.org/10.1109/MLSP49062.2020.9231926`. 39

Siddharth Sigtia, Emmanouil Benetos, and Simon Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE ACM Trans. Audio Speech Lang. Process.*, 24(5):927–939, 2016. doi:10.1109/TASLP.2016.2533858. URL `https://doi.org/10.1109/TASLP.2016.2533858`. 21, 24, 25, 49, 51, 53, 55

Magnus Själander, Magnus Jahre, Gunnar Tufte, and Nico Reissmann. EPIC: An energy-efficient, high-performance GPGPU computing research infrastructure, 2019. iii, 40

P. Smaragdis and J. C. Brown. Non-negative matrix factorization for polyphonic music transcription. In *2003 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (IEEE Cat. No.03TH8684)*, pages 177–180, 2003. doi:10.1109/ASPAA.2003.1285860. 20, 21

Xuchen Song, Qiuqiang Kong, Xingjian Du, and Yuxuan Wang. Catnet: music source separation system with mix-audio augmentation. *CoRR*, abs/2102.09966, 2021. URL `https://arxiv.org/abs/2102.09966`. 27

Daniel Stoller, Sebastian Ewert, and Simon Dixon. Wave-u-net: A multi-scale neural network for end-to-end audio source separation. In Emilia Gómez, Xiao Hu, Eric Humphrey, and Emmanouil Benetos, editors, *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*, pages 334–340, 2018. URL `http://ismir2018.ircam.fr/doc/pdfs/205_Paper.pdf`. iii, 26

Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix - A reference implementation for music source separation. *J. Open Source Softw.*, 4(41): 1667, 2019a. doi:10.21105/joss.01667. URL `https://doi.org/10.21105/joss.01667`. 25

Fabian-Robert Stöter, Stefan Uhlich, Antoine Liutkus, and Yuki Mitsufuji. Open-unmix - A reference implementation for music source separation. *J. Open Source Softw.*, 4(41): 1667, 2019b. doi:10.21105/joss.01667. URL `https://doi.org/10.21105/joss.01667`. 36, 49

L. Su and Y. Yang. Combining spectral and temporal representations for multipitch estimation of polyphonic music. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(10):1600–1612, 2015. doi:10.1109/TASLP.2015.2442411. 20

John Thickstun, Zaïd Harchaoui, and Sham M. Kakade. Learning features of music from scratch. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=rkFBJv9gg`. 16

Appendix

A. Stems with errors in Slakh

During the training and evaluation on tracks in the Slakh dataset, several mistakes in the dataset was found. This appendix lists all the stems that were found to have some errors. These errors has been classified into the following cases:

white-noise

Something must have gone wrong with the audio rendering–the audio for the stem only consists of white noise

wrong-pitch

The pitch of the label and audio are not the same

wrong-octave

The octave of the label and audio are not the same

missing-audio

Not all the notes in the label are rendered

short-labels

Some of the notes in the MIDI file parsed with PrettyMIDI are shorter than the rendered audio

long-labels

Some of the notes in the MIDI file parsed with PrettyMIDI are longer than the rendered audio

List on the next pages

```
{
    "Track00262": {
        "S01": "short-labels"
    },
    "Track00357": {
        "S03": "white-noise"
    },
    "Track00377": {
        "S07": "white-noise"
    },
    "Track00385": {
        "S00": "white-noise"
    },
    "Track00398": {
        "S00": "white-noise"
    },
    "Track00400": {
        "S00": "white-noise"
    },
    "Track00404": {
        "S03": "long-labels"
    },
    "Track00496": {
        "S01": "wrong-pitch"
    },
    "Track00629": {
        "S01": "white-noise"
    },
    "Track00633": {
        "S01": "white-noise"
    },
    "Track00737": {
        "S01": "long-labels"
    },
    "Track00749": {
        "S01": "white-noise"
    },
    "Track00893": {
        "S01": "long-labels"
    },
    "Track01629": {
        "S00": "white-noise"
    },
    "Track01876": {
        "S01": "missing-audio"
    },
    "Track01908": {
        "S05": "missing-audio"
    },
    "Track01918": {
        "S10": "wrong-pitch"
    },
    "Track01929": {
        "S04": "wrong-octave"
    },
    "Track01931": {
        "S01": "wrong-pitch"
    },
    "Track01993": {
        "S01": "missing-audio"
    },
    "Track01937": {
        "S03": "wrong-pitch"
    },
    "Track02024": {
        "S13": "missing-audio"
    }
}
```

B. Additional Results

Following are a collection of transcriptions from the Slakh test set to show the performance of the models and to help get an intuition for the transcription metrics. Each figure are around five seconds of audio. The upper half shows the models prediction with different colors. Blue comes from the onset detector head, purple from the frame detector and yellow offset from the offset detector. Black in the lower half are correct prediction, orange are incorrect prediction and light blue are missed prediction. Note onsets are in a darker color in the lower half. If viewed on a display, an inversion of the colors are recommended. In the caption for each figure, the experiment, track and frame F_1 score, note F_1 and note-with-offset F_1 are shown. These metrics are from the shown segment, not the whole track.

Figures on the next pages

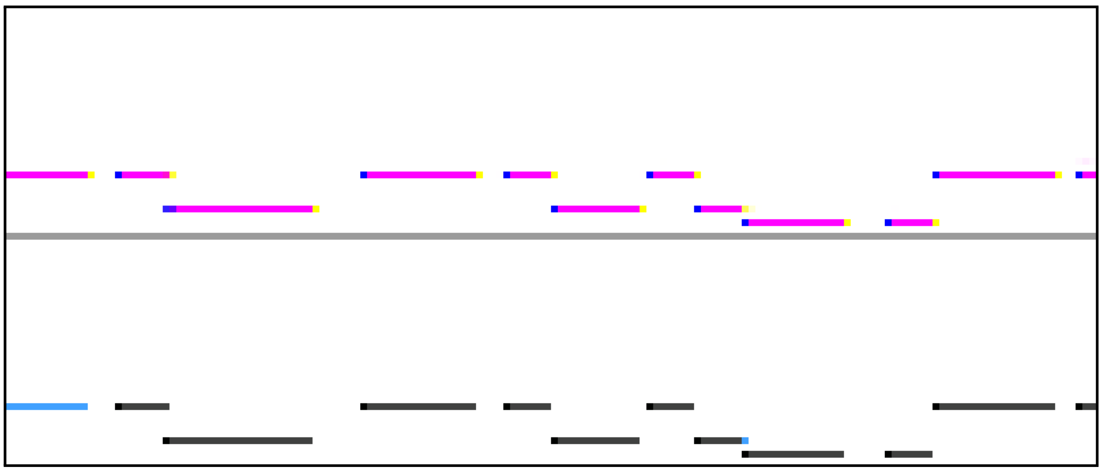

Figure 1.: Experiment 0a, Track01881, 0.996|1.000|1.000

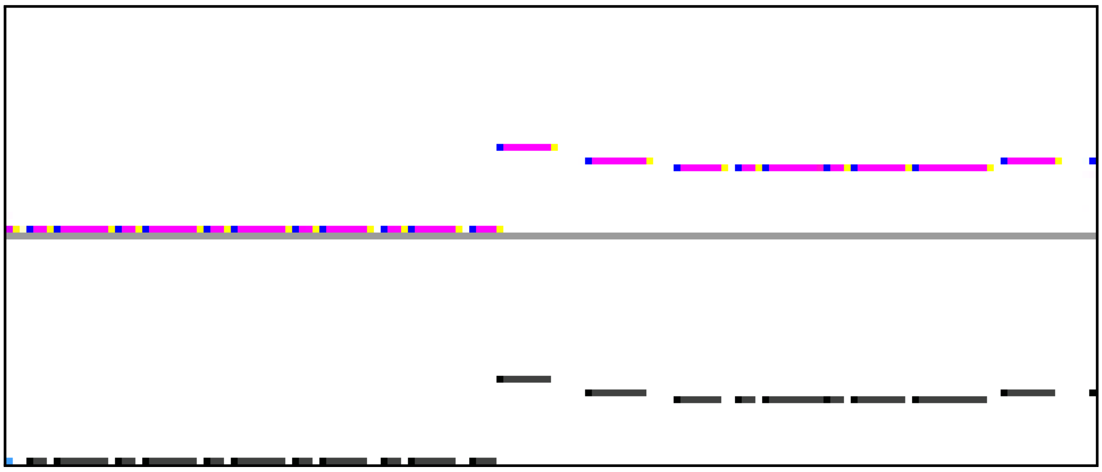

Figure 2.: Experiment 0a, Track01892, 1.000|1.000|1.000

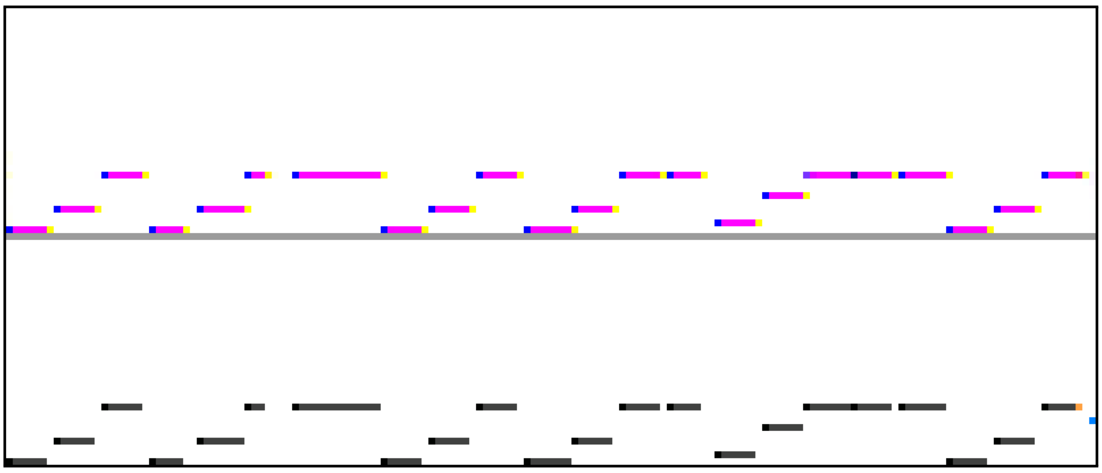

Figure 3.: Experiment 0a, Track01895, 0.993|0.978|0.978

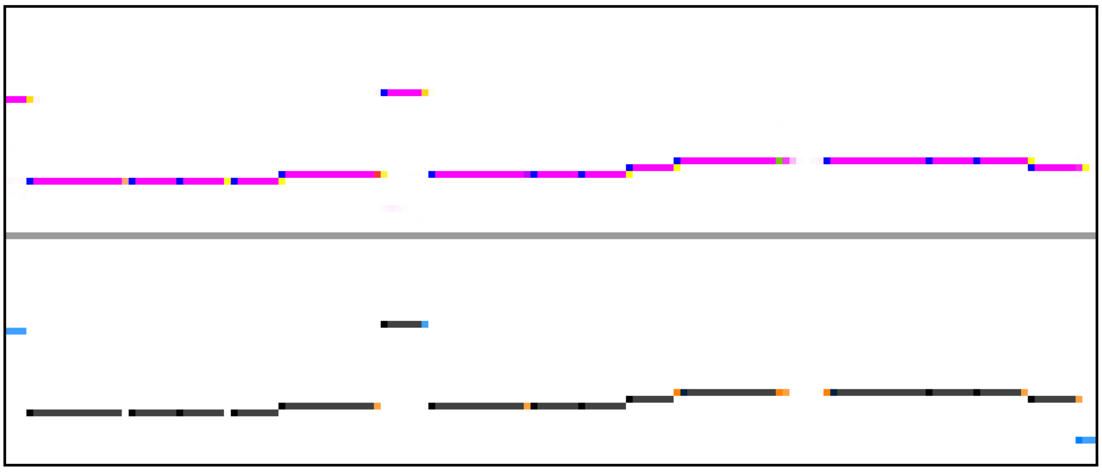

Figure 4.: Experiment 0a, Track01901, 0.959|0.938|0.875

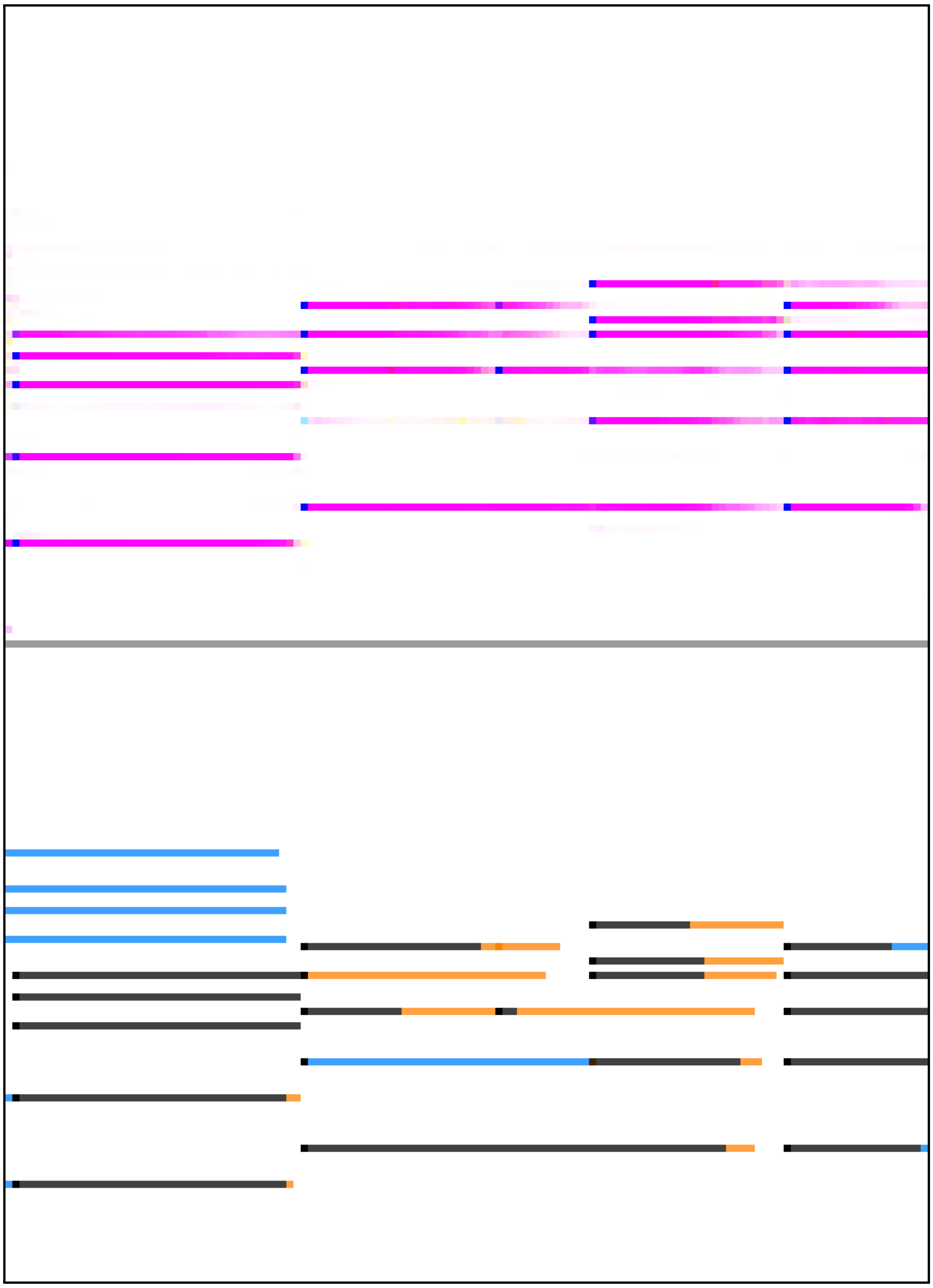

Figure 5.: Experiment 0b, Track01878, 0.837|0.950|0.450

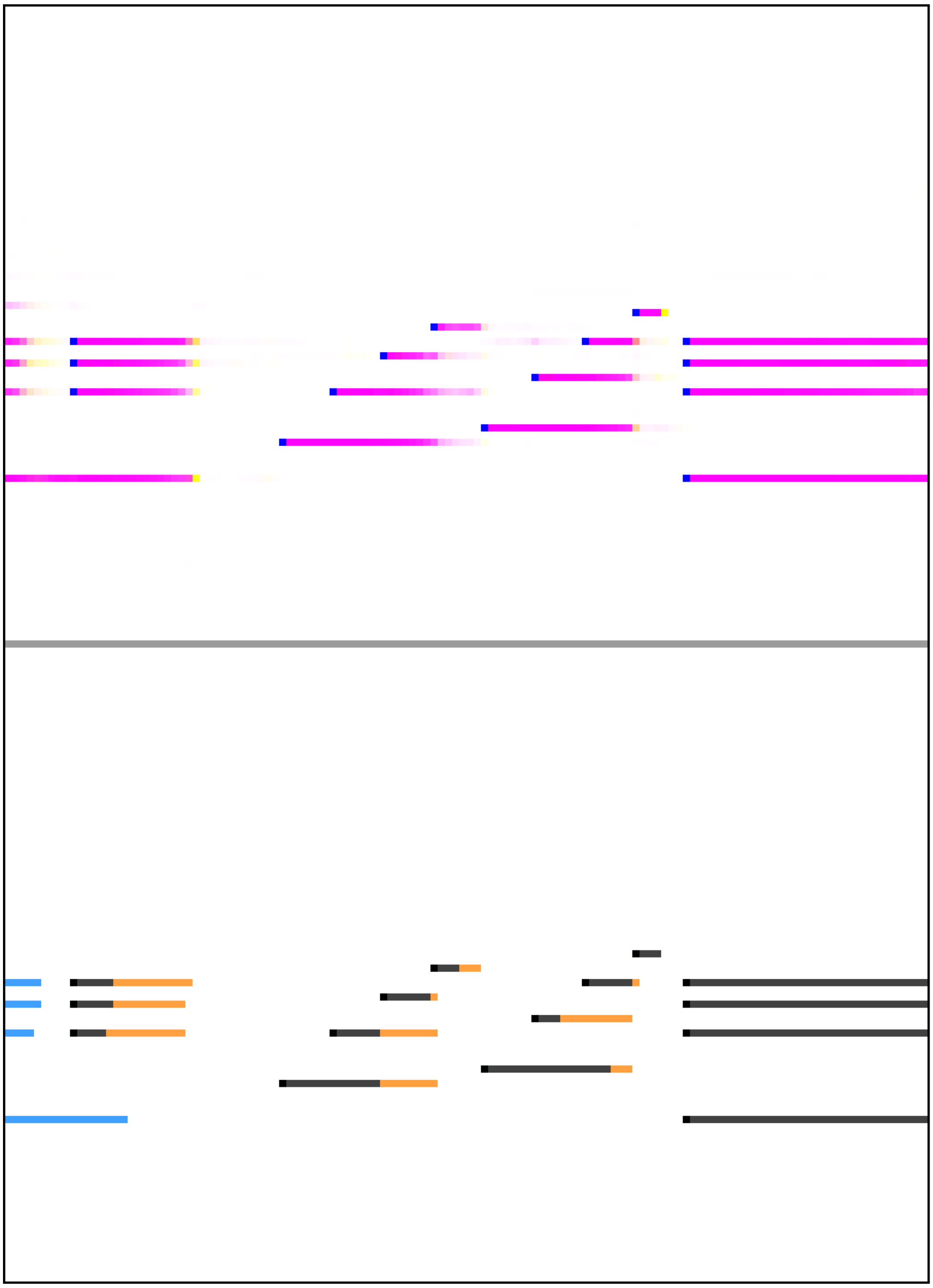
Figure 6.: Experiment 0b, Track01881, 0.869|1.000|0.533

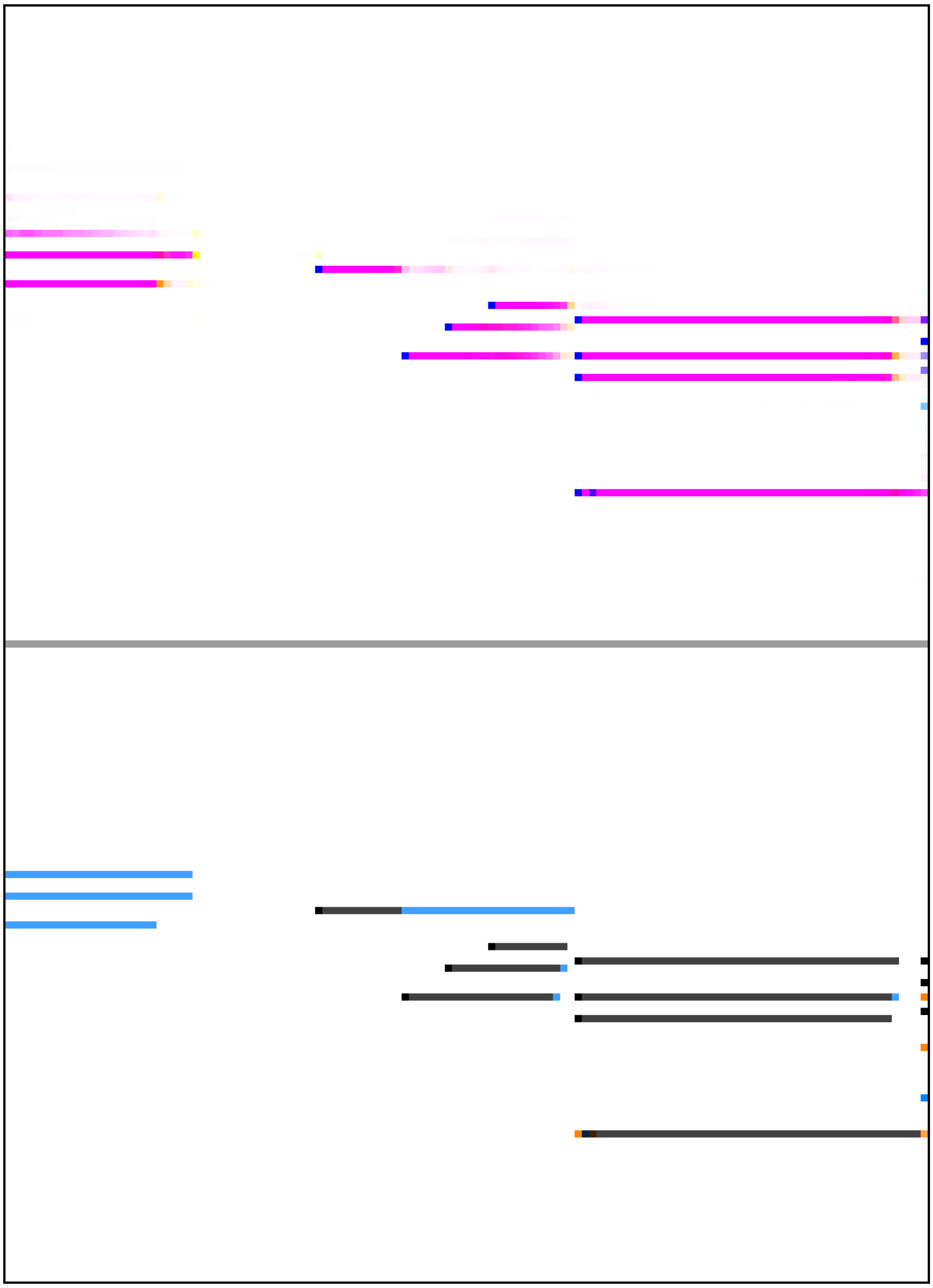
Figure 7.: Experiment 0b, Track01888, 0.938|0.846|0.769

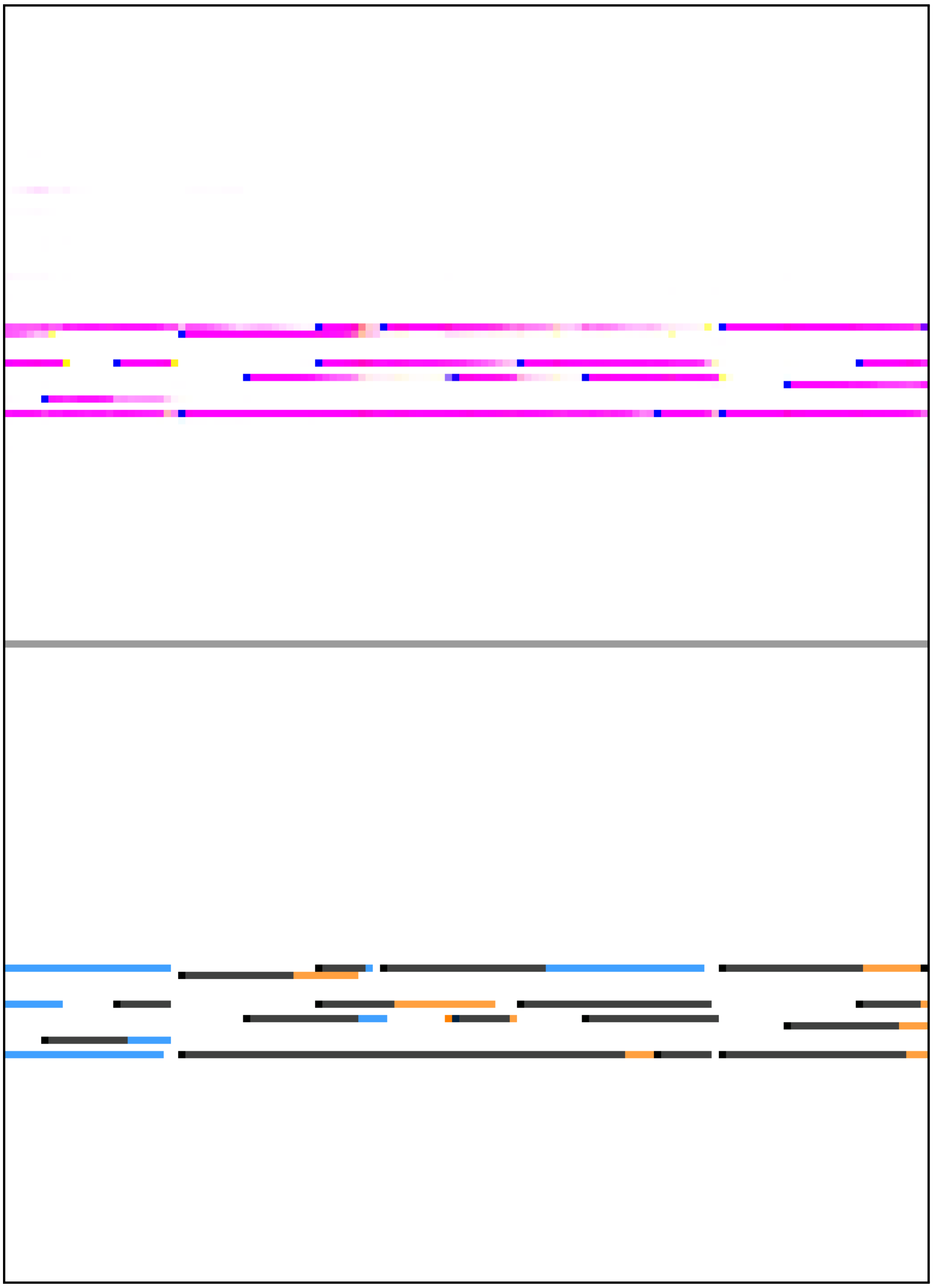
Figure 8.: Experiment 0b, Track01889, 0.881|1.000|0.588

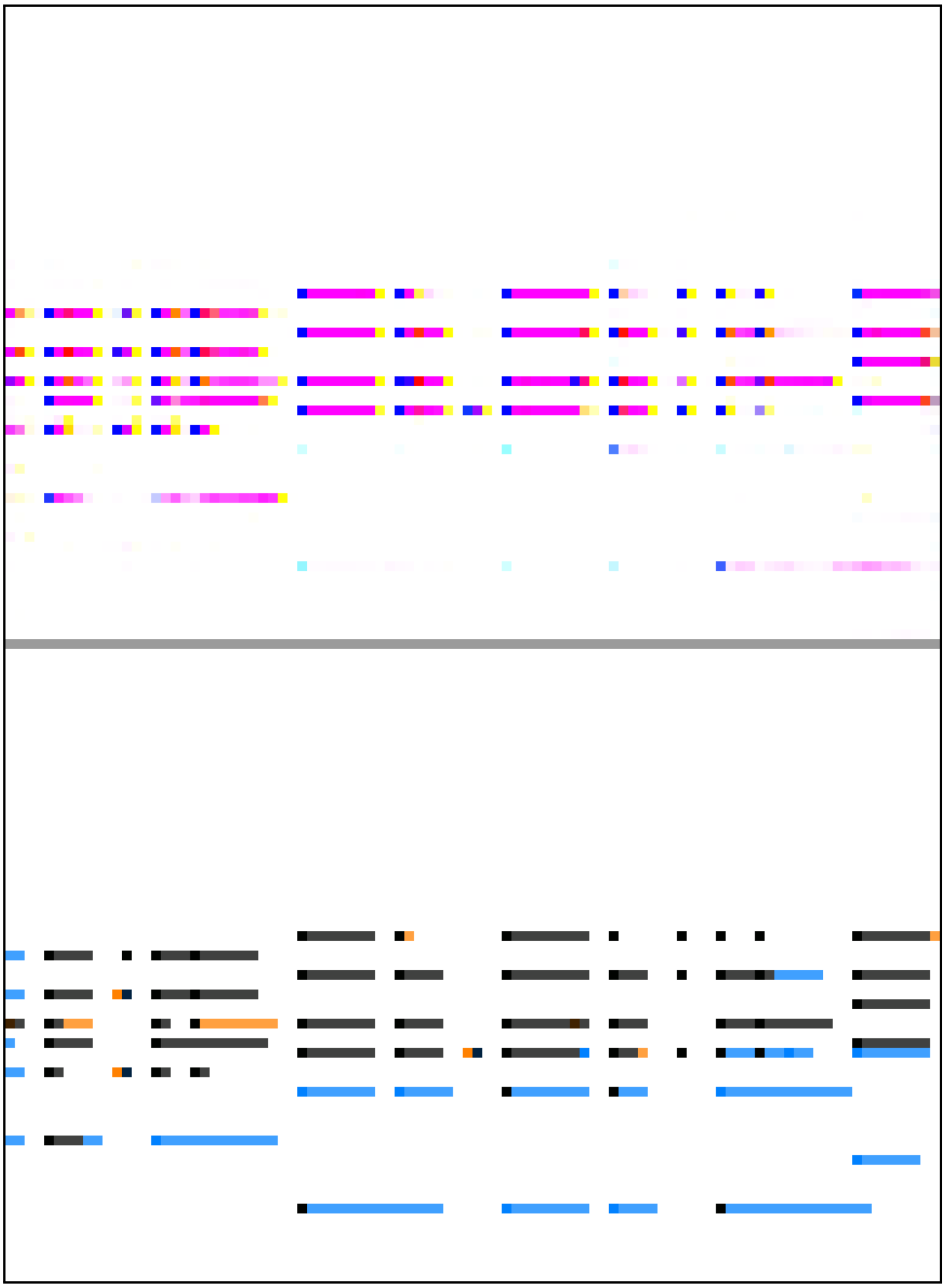
Figure 9.: Experiment 0c, Track01877, 0.755|0.900|0.717

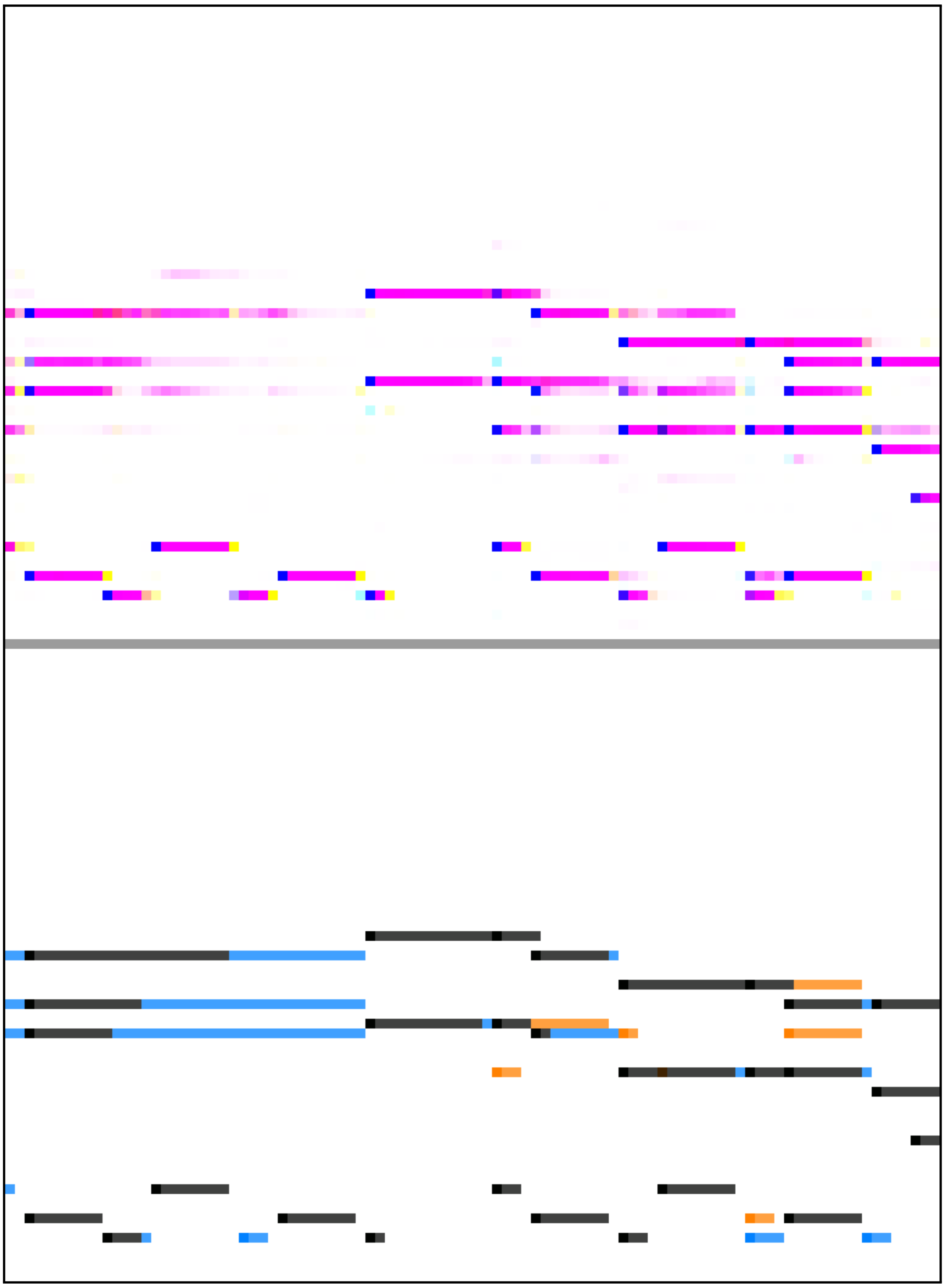
Figure 10.: Experiment 0c, Track01892, 0.785|0.875|0.562

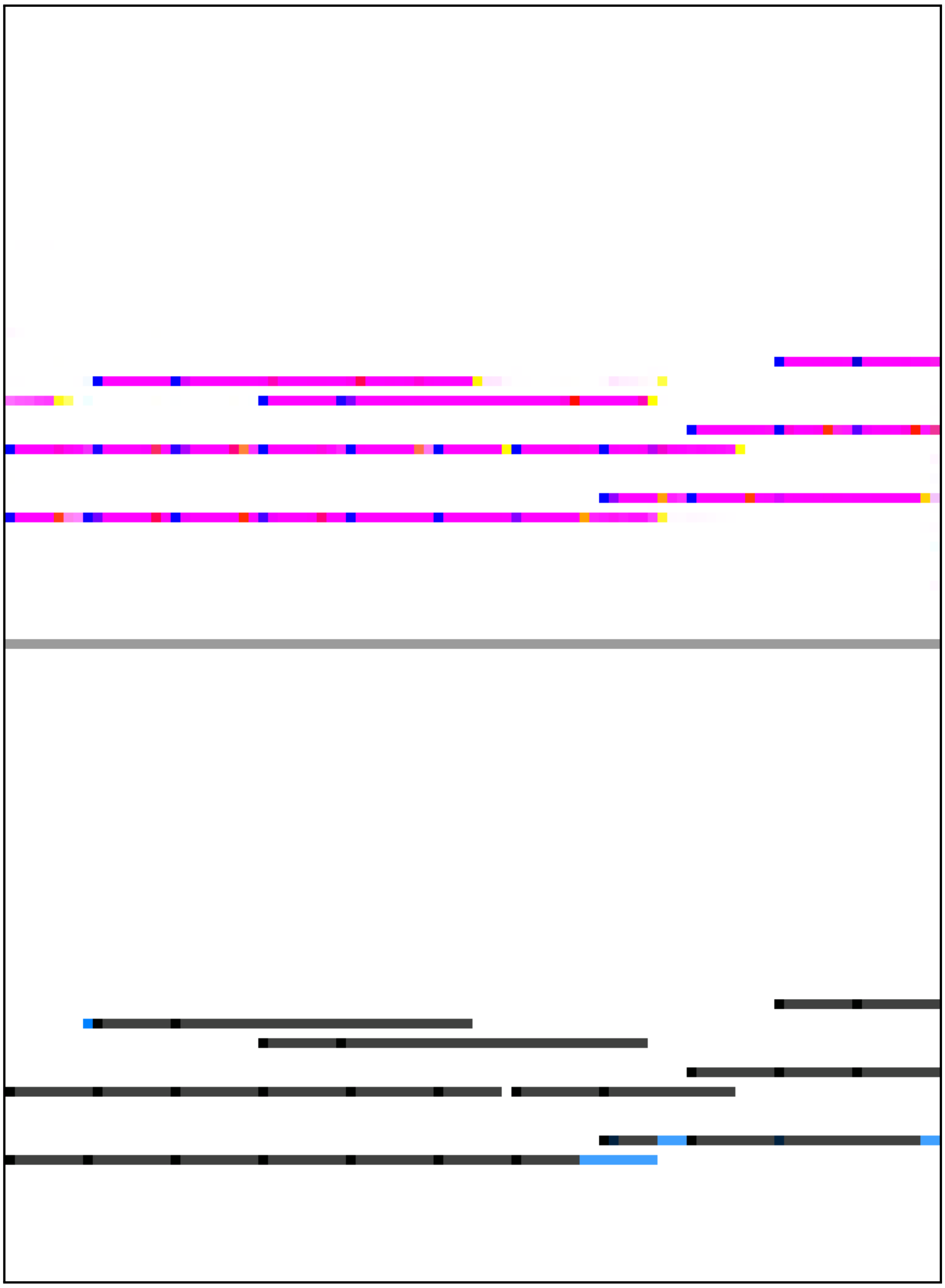

Figure 11.: Experiment 0c, Track01893, 0.976|0.981|0.830

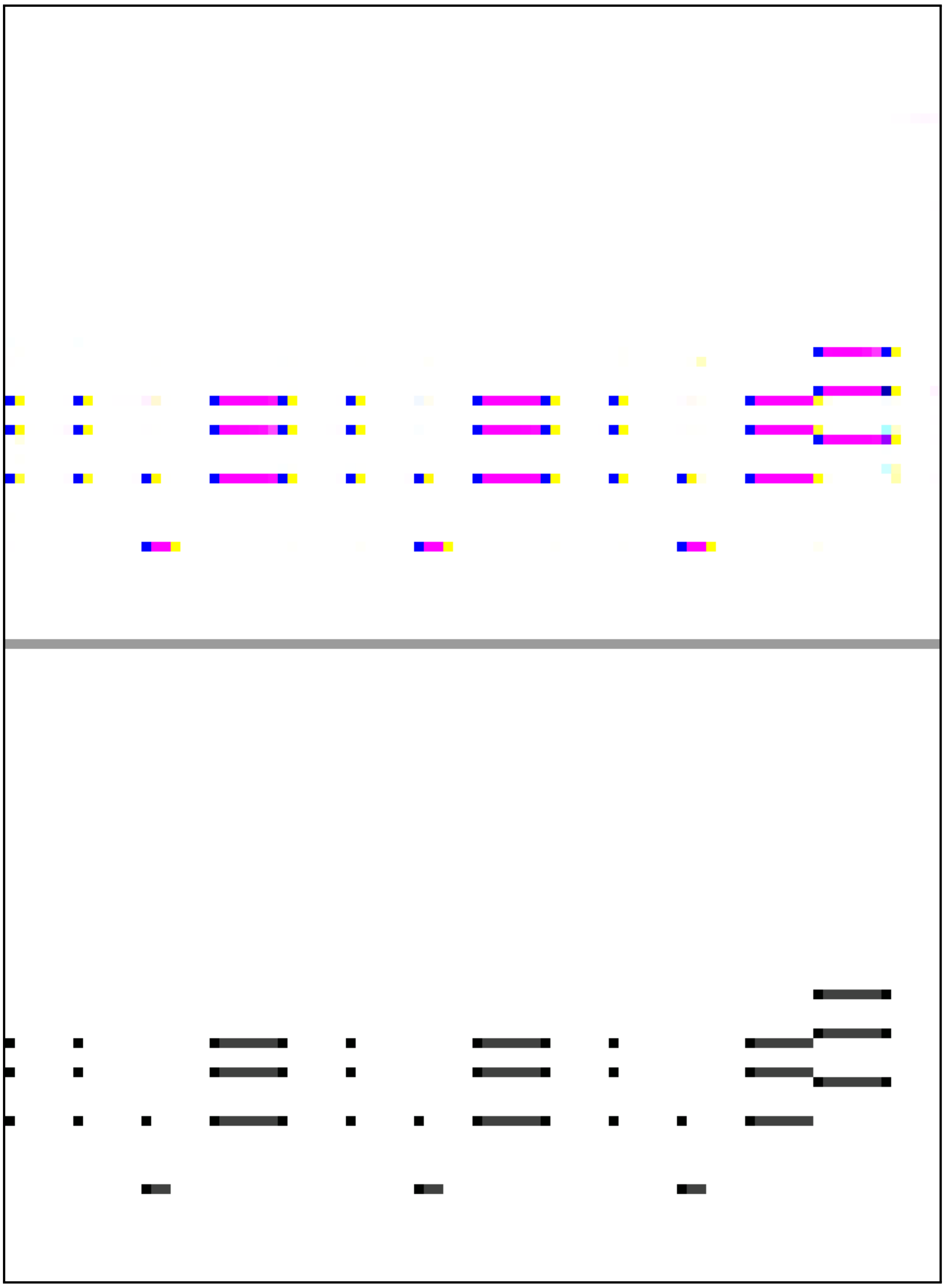

Figure 12.: Experiment 0c, Track01895, 1.000|1.000|1.000

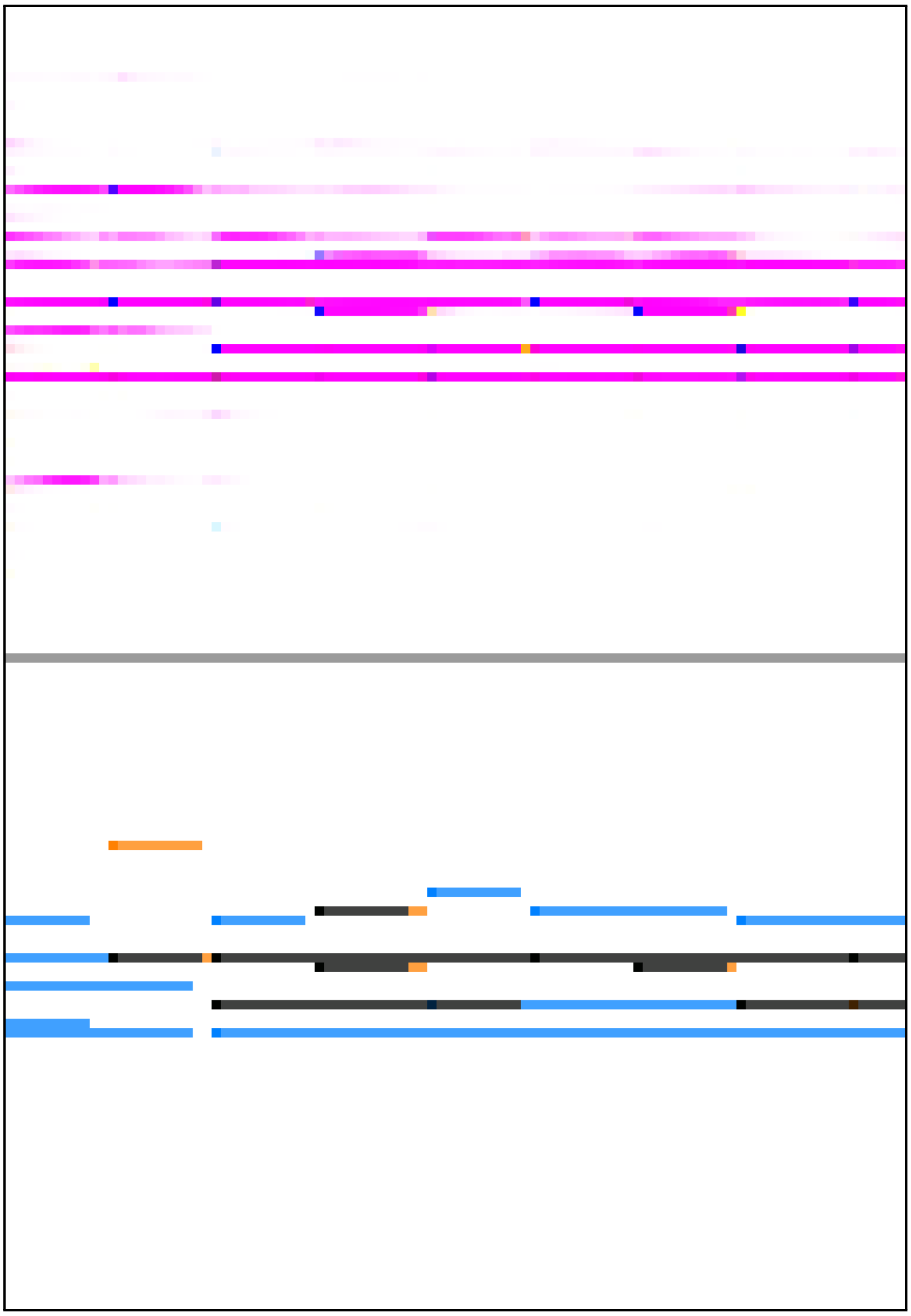
Figure 13.: Experiment 4a, Track01882, 0.657|0.692|0.538

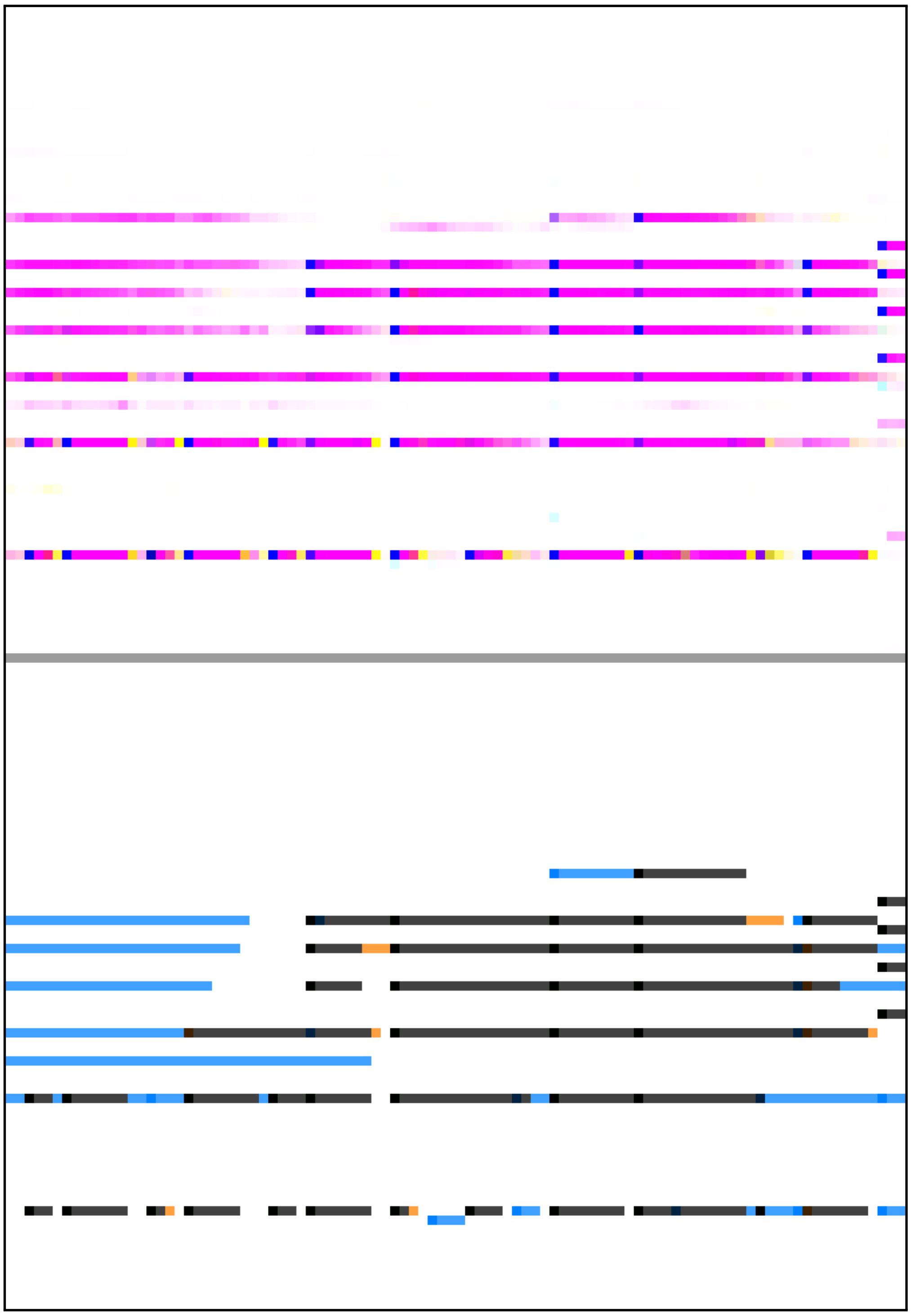

Figure 14.: Experiment 4a, Track01892, 0.900|0.889|0.626

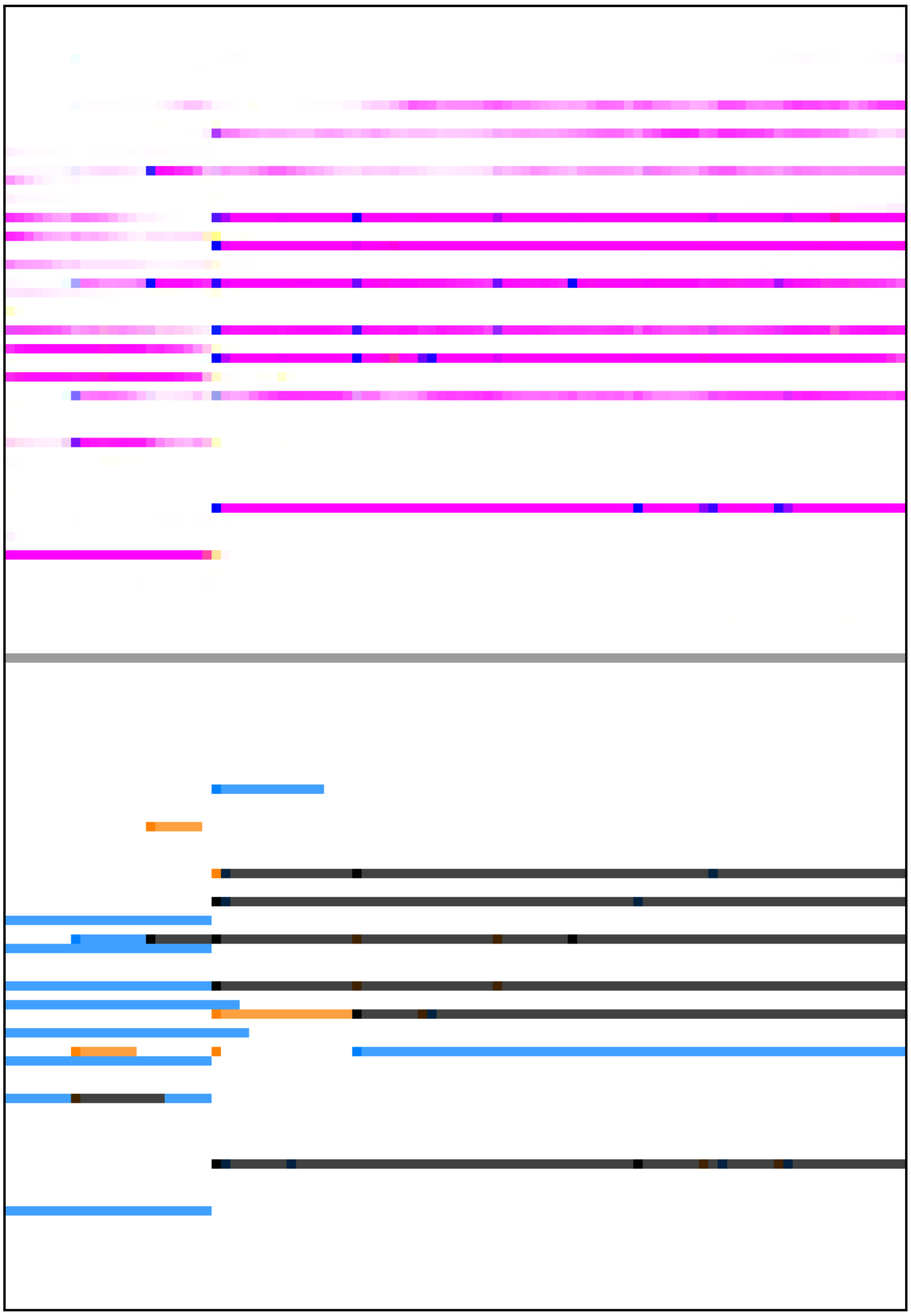

Figure 15.: Experiment 4a, Track01932, 0.880|0.585|0.585

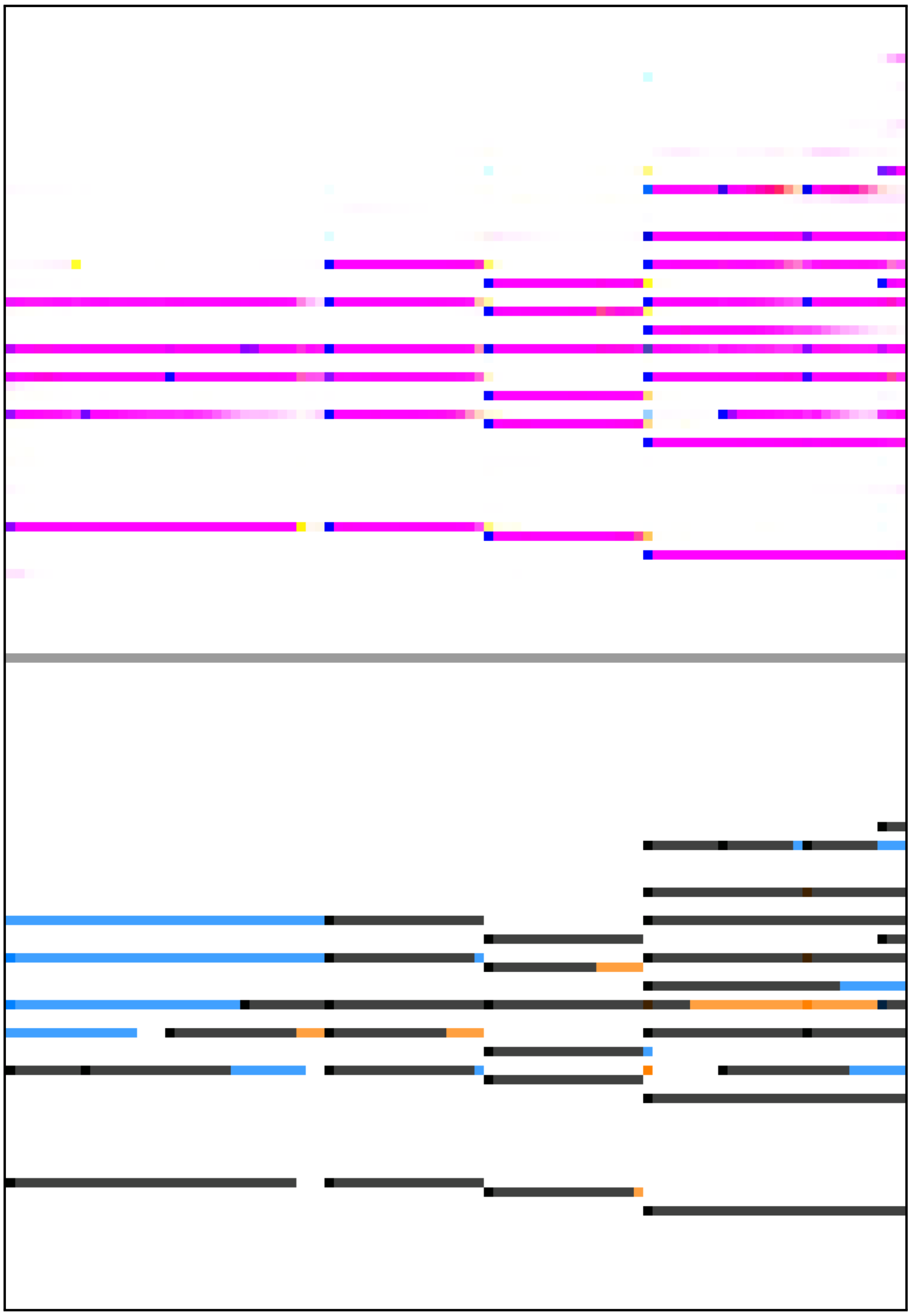

Figure 16.: Experiment 4a, Track01950, 0.894|0.886|0.514

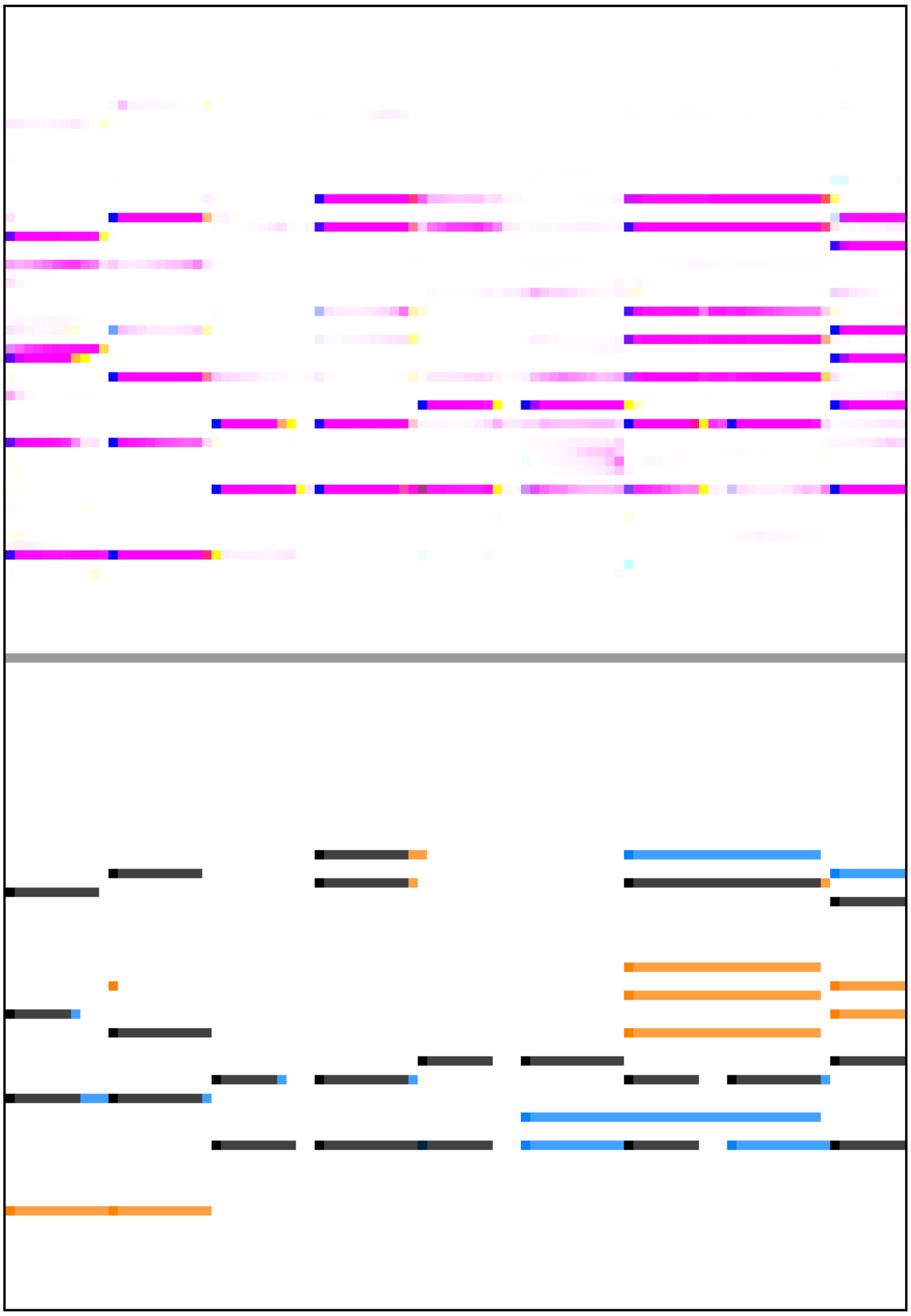

Figure 17.: Experiment 4a, Track01955, 0.682|0.750|0.714

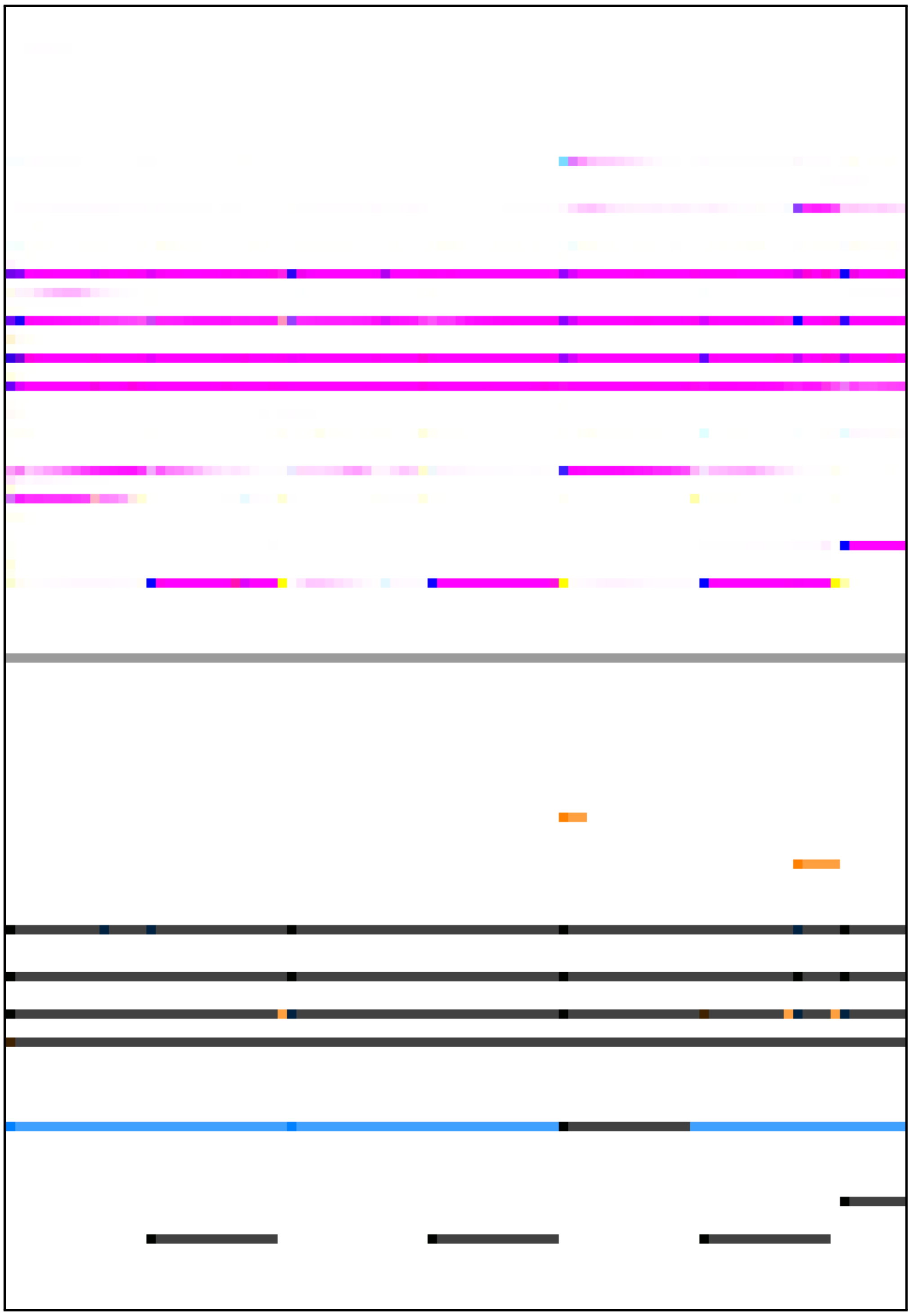

Figure 18.: Experiment 4a, Track01956, 0.786|0.727|0.591

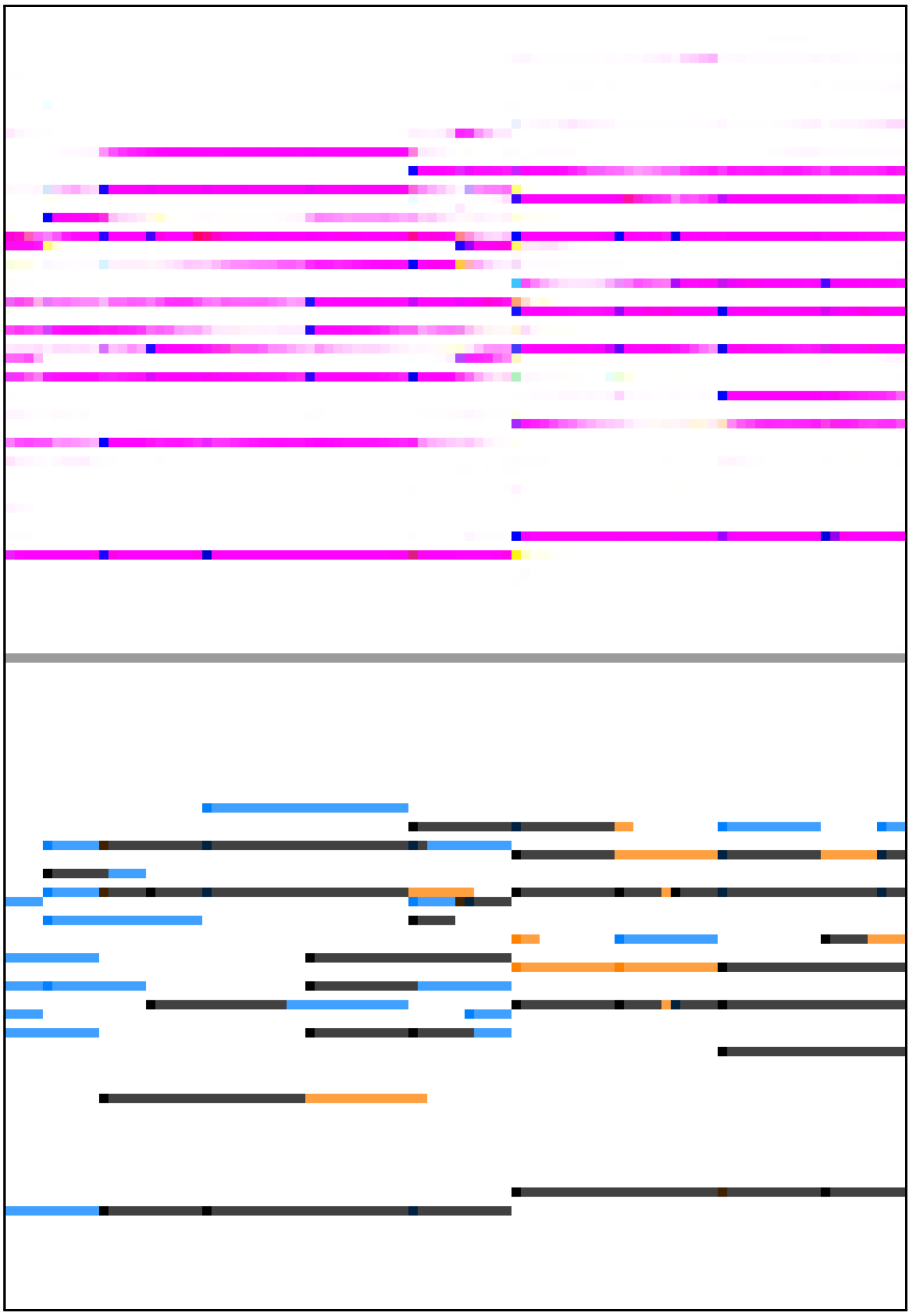

Figure 19.: Experiment 4a, Track01957, 0.808|0.658|0.342

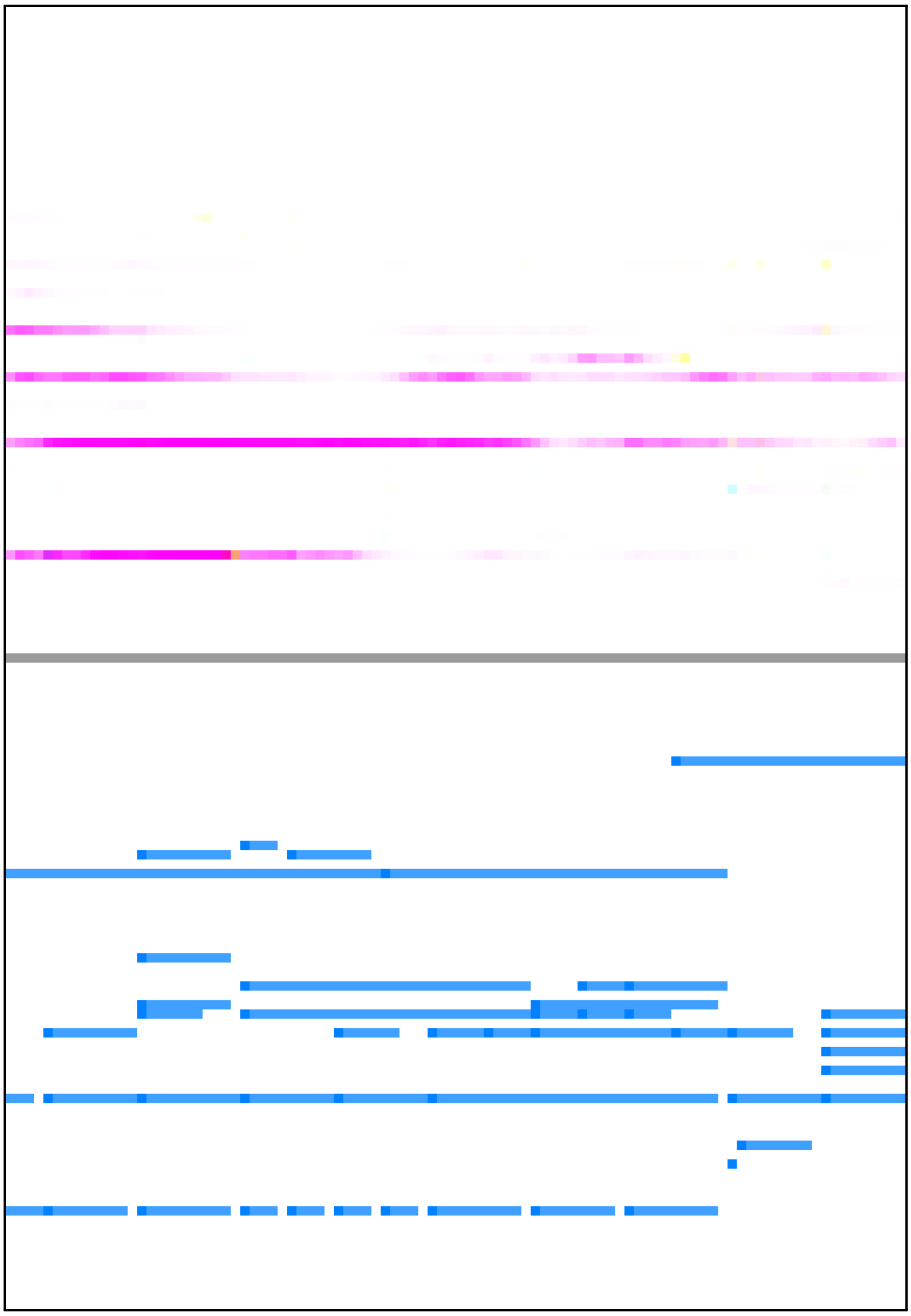

Figure 20.: Experiment 4a, Track01959, 0.000|0.000|0.000

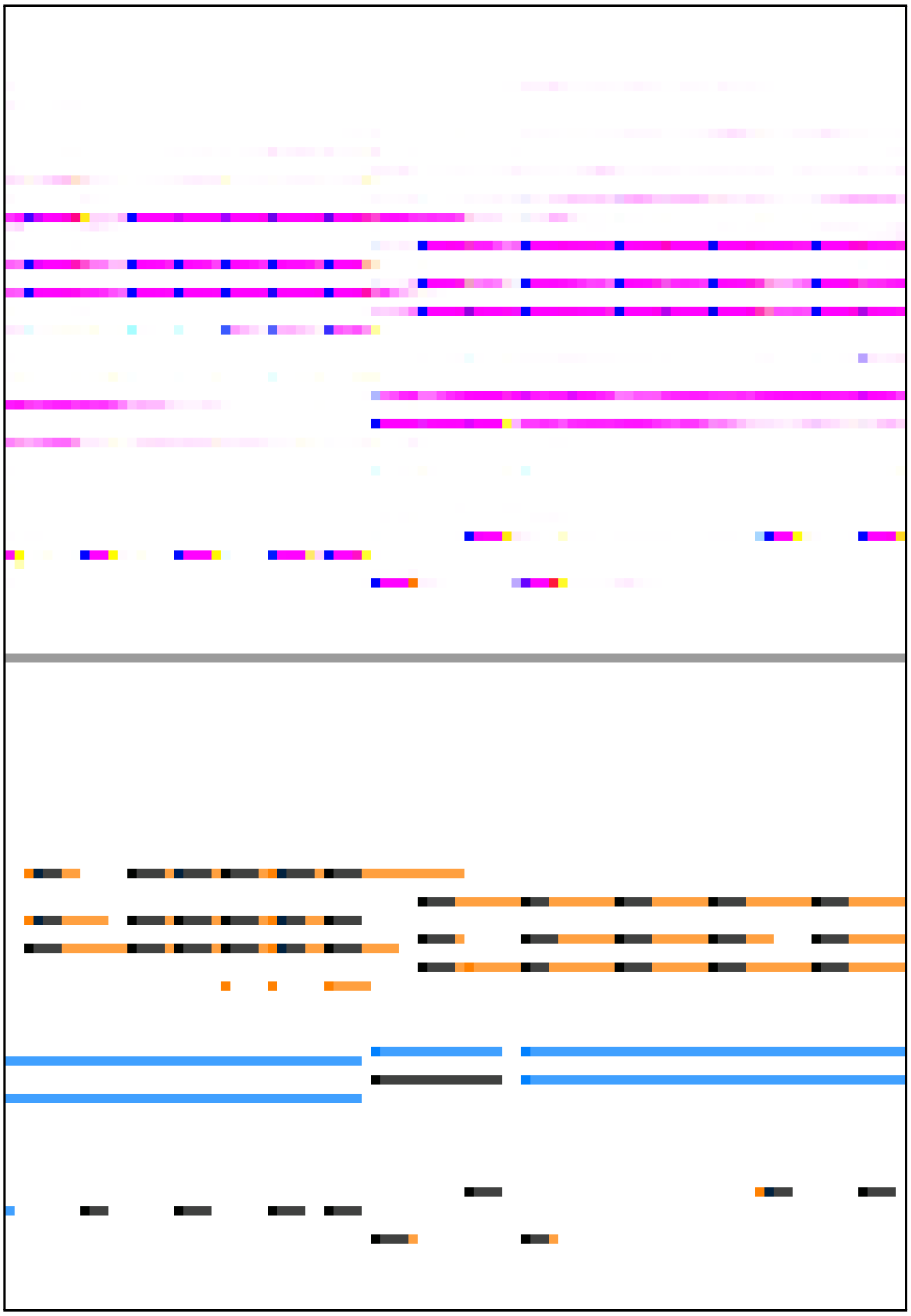

Figure 21.: Experiment 4a, Track01963, 0.588|0.913|0.261

Henrik Grønbech

Multi-Instrument Automatic Music Transcription with Deep Learning