Karlsruher Institut für Technologie

Automatic music transcription using sequence to sequence learning

Master's thesis of

B.Sc. Maximilian Awiszus

At the faculty of Computer Science
Institute for Anthropomatics and Robotics

Reviewer:	Prof. Dr. Alexander Waibel
Second reviewer:	Prof. Dr.
Advisor:	M.Sc. Thai-Son Nguyen

Duration: 25. Mai 2019 – 25. November 2019

Interactive Systems Labs
Institute for Anthropomatics and Robotics
Karlsruhe Institute of Technology
Title: Automatic music transcription using sequence to sequence learning
Author: B.Sc. Maximilian Awiszus

Maximilian Awiszus
Kronenstraße 12
76133 Karlsruhe
maximilian.awiszus@student.kit.edu

Statement of Authorship

I hereby declare that this thesis is my own original work which I created without illegitimate help by others, that I have not used any other sources or resources than the ones indicated and that due acknowledgement is given where reference is made to the work of others.

Karlsruhe, 15. März 2017

.....
(B.Sc. Maximilian Awiszus)

Contents

Abstract

1. Introduction

Automatic music transcription (AMT) is a term used in the field of music information retrieval (MIR). In this interdisciplinary field AMT is generally defined as a process of converting acoustic music in a symbolic musical notation as depicted in fig. 1.1. On the left in the figure is a sound wave symbolising the the acoustic musical signal and the right sheet music is printed as an exemplary symbolic musical notation.

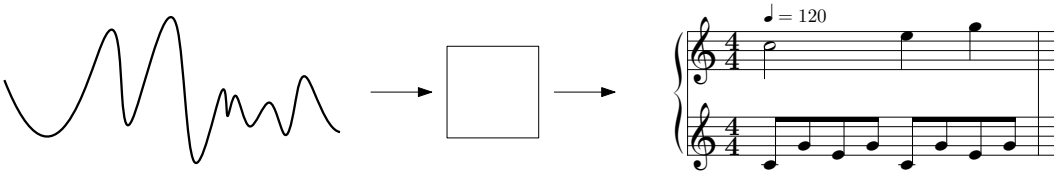


Figure 1.1: Automatic music transcription

AMT itself is the key to music information retrieval where disciplines of music, psychology, computer science, engeneering science and information science come together. Acoustic music is converted in a format which is easier to analyse by human as well as automatically analysable by machines. An AMT system helps music students analysing a music piece where a written form of this piece is missing. In addition to that it makes it easier for student even to analyse longer music pieces where a transcription by ear would take to much time. Also a transcribed version can be used to give student feedback what they humanly transcribed wrong and can help to train the ability of transcribing by ear. Furthermore composers get support and can convert music into a notational form automatically. This helps the composers to concentrate on the creative part of playing musical sounds instead of thinking about the specific notation. Also for a band rehearsal it could be useful when a musician plays music and can directly show her or his colleagues the sheet music to play along or share the musical idea with other bands.

More precisely AMT is defined by the non-profit organization International Society for Music Information Retrieval (ISMIR): Each year at the symposium hosted by the ISMIR algorithms in the field of MIR are evaluated as part of the Music Information Retrieval

Evaluation eXchange (MIREX). Music transcription within this scope is defined as *MIREX multiple-F0* task where each estimated note contains information of the appropriate frequency and the time information. The temporal information is divided in the note on- and the note offset.

To understand the underlying approach different specialised terminology resp. relationships need to be introduced. In the following a briefly introduction into acoustic music (section 1.1) and the musical notaion (section 1.2) as well as the MIDI standard (section 1.3) will be given. Also a mathematical tool (section 1.5) to analyse an inference of different instruments playing together (section 1.4) is explained. Furthermore in the latter section the already in the title mentioned sequence to sequence approach is presented (section 1.6) jointly by outlining important modules of neural networks.

1.1 Acoustic music

Acoustic music is a vibration resp. oscillation which is propagated in a medium, e. g. the air. Physically this can be described as a longitudinal wave y at position x and time t where y_0 is the amplitude, f the frequency and λ the wave length of the oscillation

$$y(x,t) = y_0 \cos\left(2\pi\left(f \cdot t - \frac{x}{\lambda}\right)\right), \quad x,t,y_0,f,\lambda \in \mathbb{R}. \tag{1.1}$$

The bigger the frequency gets the higher the acoustic music is recognised and a bigger amplitude results in a louder acoustical signal. In the following the wave is consider to be observed at a fix position. Assuming $x := 0$ we get a simpler equation denoted as

$$y(t) = y_0 \cos\left(2\pi f t\right), \quad t,y_0,f \in \mathbb{R}. \tag{1.2}$$

In fig. 1.2 three exemplary waves with different frequencies $f_1 := 1$, $f_2 := 3$, $f_3 := 5$ and amplitudes $y_{0_1} := 1/2$, $y_{0_2} := 3/4$, $y_{0_3} := 1$ can be examined.

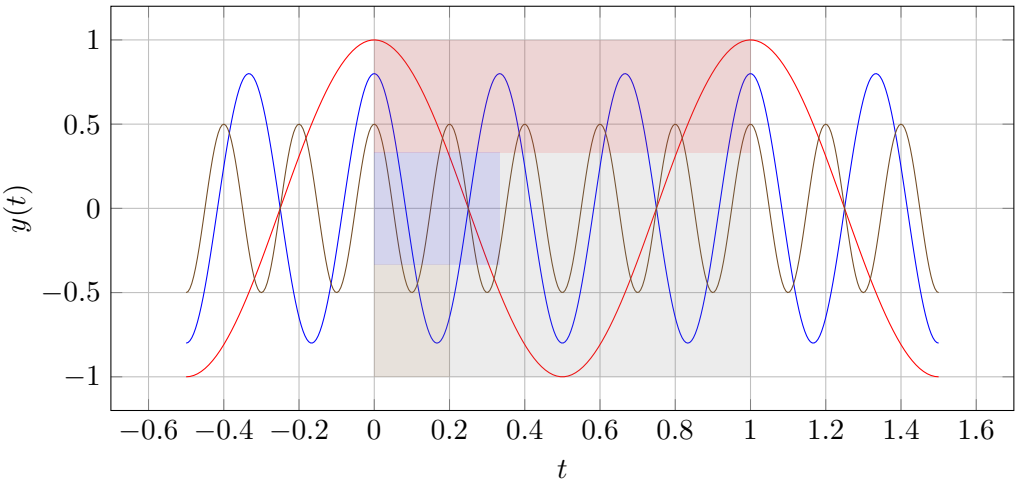


Figure 1.2: Waves with frequency $f_1 = 1$, $f_2 = 3$ and $f_3 = 5$ and different amplitudes.

1.2 Musical note and sheet music

Each musical note consists of two information namely the pitch and the duration. The pitch of a note itself is denoted as a 3-tuple $(p, o, a) \in \mathcal{P} \times \mathcal{O} \times \mathcal{A}$ of pitch class $\mathcal{P}$, a number for the octave $\mathcal{O}$ and accidentals $\mathcal{A}$. The combination of pitch class, octave and accidental correspond to a specific frequency of the note and is measured in Hertz (Hz). There exist different naming convections (e. g. Solfège[1] or Indian Sanskrit letters) for the pitch classes. In the following the pitch classes

$$\mathcal{P} := [A, B, C, D, E, F, G] \tag{1.3}$$

which are common in most English-speaking regions are used. A pitch class describes notes with similar sound which physically means that the frequencies w.l.o.g. f_1 and f_2 of the notes of a same pitch class have a ratio specified as

$$f_1 = 2^i \cdot f_2, \quad i \in \mathbb{Z} \tag{1.4}$$

Also for the already mentioned octave number $\mathcal{O}$ different conventions exist. In this thesis the *Scientific pitch notation* (SPN) will be used which means that all notes in a pitch class are numbered from 0 to 9. For example the third pitch class `C` contains ten different notes namely $[C_0, C_1, C_2, \ldots, C_9]$. Each combination of pitch class and number can be visualized as a dot on five lines which are called staff as depicted in fig. 1.3.

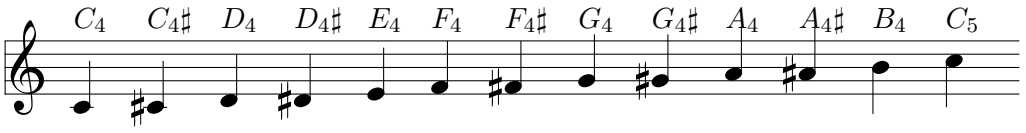


Figure 1.3: Quarter semitone notes on a staff ranging one octave from C_4 to C_5 (SPN)

The accidental $\mathcal{A}$ of a note can be a sharp $\sharp$ or flat sign $\flat$. The $\sharp$ increases the note by a semitone which is a frequency ratio of $\sqrt[12]{2} \approx 1.0595$. Similarly the $\flat$ decreases a note by the same ratio.

The smallest step in the twelve-tone equal temperament[2] is a semitone and thus the accidental creates ambiguous names for exactly one note, e. g. $C_4\sharp = D_4\flat$. If the natural note without increase or decrease is meant the accidental will be left of or the $\natural$ symbol is used. One above mentioned octave covers 12 semitones which also explains the name of the musical temperament.

Also for the mapping from notes to frequencies several standards exist. The most popular tuning is called *Concert pitch* and assigns $A_4 := 440\,\text{Hz}$.

A note can have different discrete durations. In fig. 1.3 all notes are quarters. Commonly used durations are sixteenth, eighth, quarter, half or whole (c. f. table 1.1). All this notations refer to the current tempo of the music piece which is given in beats per minute (bpm). This tempo could be assign to different note durations. Assumed in the beginning of a music piece the tempo is assigned to a quarter, e. g. $\text{♩} = 120$ means that one quarter note takes 0.5 s and thus e. g. a whole takes 2 s.

[1] used in Italy, Spain, France, ... $\mathcal{P}_{\text{Solfège}} := [\text{Do}, \text{Re}, \text{Mi}, \text{Fa}, \text{Sol}, \text{La}, \text{Si}]$

[2] The twelve-tone equal temperament is the today's most common musical temperament.

Notation	Name	Duration
𝅘𝅥𝅯	Sixteenth	1/16
𝅘𝅥𝅮	Eighth	1/8
𝅘𝅥	Quarter	1/4
𝅗𝅥	Half	1/2
𝅝	Whole	1

Table 1.1: Some discrete duration of musical notes

Using all this conventions the so called *sheet music* which is the most common musical notations can be understood. In fig. 1.4 a excerpt of the sheet music of Mozart's sonata in C-major can be observed. Whereas the two staffs in the figure over each other are two note sequences which are played in parallel.

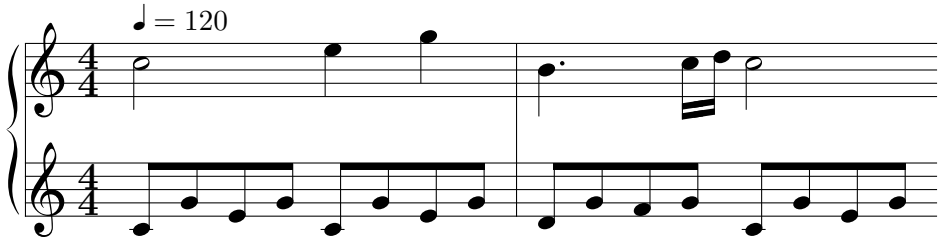


Figure 1.4: Excerpt of the sheet music of Mozart's sonata in C-major

1.3 Musical Instrument Digital Interface

Another way of describing music is the Musical Instrument Digital Interface (MIDI) standard which has been incorporated in 1983. It is commonly used for communication between different electronic music devices, e. g. keyboards and personal computers. In this standard all the above mentioned notes are numbered with integer numbers ranging from 0 to 127. Taking the context of section 1.2 the relationship between the frequency f of a note and the MIDI number n_{MIDI} can be described as

$$f(n_{\text{MIDI}}) = \left(\sqrt[12]{2}\right)^{70-n_{\text{MIDI}}} \cdot 440\,\text{Hz}, \quad n_{\text{MIDI}} \in [0, 127]. \tag{1.5}$$

The A_4 gets the MIDI number 69 and is marked in fig. 1.5 with a blue triangle.

The equivalent to sheet music is realised by a sequence of different events called MIDI messages. Indicating a note to be played, e. g. if a piano key is pressed a `NOTE_ON` message is sent as described in the MIDI 1.0 standard. If the note stops resp. the piano key is released a `NOTE_OFF` message is sent. In the MIDI standard there exists 16 different channels which can for example be used for different instruments. As depicted in fig. 1.6

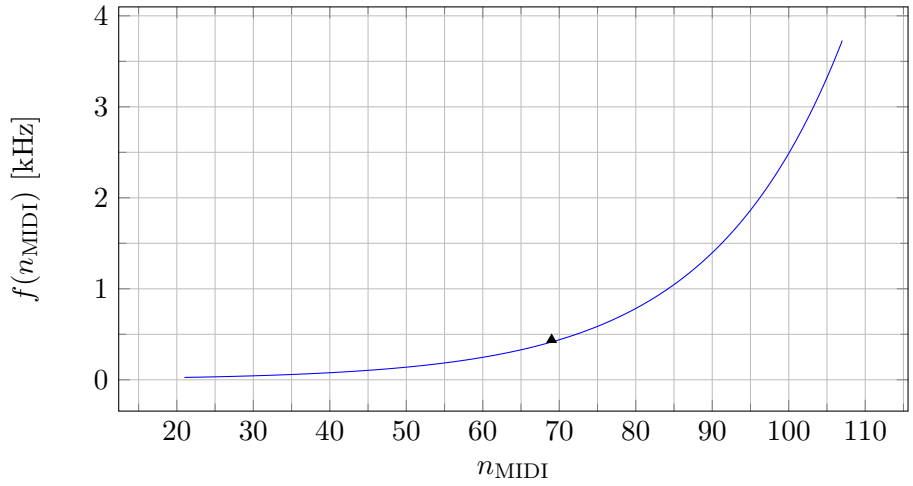


Figure 1.5: Relation between note frequency and the midi number

`NOTE_OFF`	`0x80 - 0x8F`	Note number	Velocity
`NOTE_ON`	`0x90 - 0x9F`		

Figure 1.6: 3 bytes of two MIDI 1.0 messages

in column two to four these two events contain three bytes: the event number and an offset for the channel, the note number and the velocity of the on- resp. off-set.

If the events of `NOTE_ON` and `NOTE_OFF` are sent at discrete points of time a sequence of notes which could be sheet music can be graphically visualised as a piano roll (c. f. fig. 1.7).

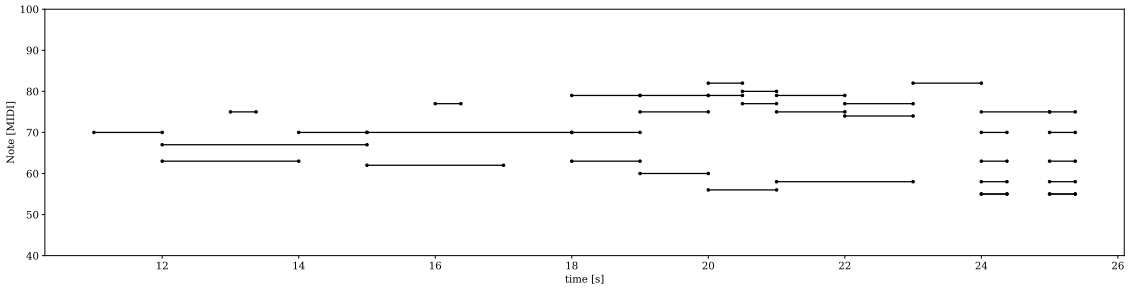


Figure 1.7: Piano roll

1.4 Instruments and inference

If two waves resonante simultaneously both are mathematically added resp. physically an inference of these waves happens, resulting in one new wave. If the waves from fig. 1.2 are played simultaneously the inferred wave in fig. 1.8 will be the result. This occurs in

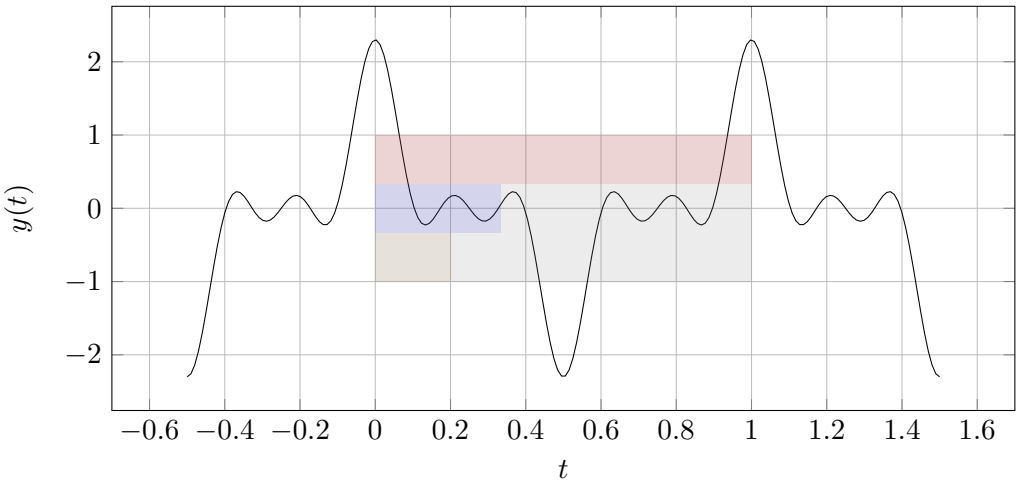


Figure 1.8: Plotted inference of waves from fig. 1.2.

monophonic music as well as in polyphonic music. If more than one note is played at a time the music is called polyphonic[3] otherwise monophonic.

Assuming an instrument is playing one note not exactly one frequency will resonate. A note can be seen as a combination of different waves called *partials*. The in section 1.2 assigned frequencies to notes are more precisely described as fundamental frequencies f_0 which are one frequency of the whole frequency mixture. The partials of the most pitched acoustic instruments can be described as harmonics which are defined as

$$\mathcal{F}_{\text{harmonics}} := \{f_0 \cdot i \mid i \in \mathbb{N}_{>0}\} \tag{1.6}$$

which means that the harmonic are integer multiples of the fundamental frequency.

1.5 Fourier analysis

A mathematical tool to analyse the frequency occurring in a inferred wave mixture as in fig. 1.8 (which could be seen as a visualised audio file) is the Fourier analysis. There exist different variants of the Fourier analysis. Assuming an audio file which is time discretely stored on a storage medium the Discrete Fourier transform (DFT) is adequate. In the following the signal is represented as a sequence (x_N) with length N. For a Fourier analysis the signal is considered to be periodically which is not the case if we analyse for example a succession of notes or a music recording.

Thus the Short-Time-Fourier-Transformation (STFT) is used instead. This transformation additionally involves a window which zero-fills values of the sequence which are not of interest to analysis for occurring frequencies. Furthermore the observed part of the sequence is assumed periodic. For $n \in [0, N]$ where N states the length of the window (also called *window size*) in the following three common windows are listed. The simple rectangular window w_R also known as Dirichlet window is defined as

$$w_R[n] := 1, \tag{1.7}$$

[3]Polyphonic music does not imply that there have to be more instruments involved, e. g. a piano, viola or guitar can produce polyphonic music.

whereas the Hann window w_H is is calculated by

$$w_H[n] := 0.5 \left[1 - \cos\left(\frac{2\pi n}{N}\right)\right] \tag{1.8}$$

and the Blackman-Harris window w_{BH} denoted as[4]

$$w_{BH}[n] := a_0 - a_1 \cos\left(\frac{2\pi n}{N}\right) + a_2 \cos\left(\frac{4\pi n}{N}\right) - a_3 \cos\left(\frac{6\pi n}{N}\right). \tag{1.9}$$

In fig. 1.9 the above defined windows w_R, w_H and w_{BH} can be examined in a plotted version.

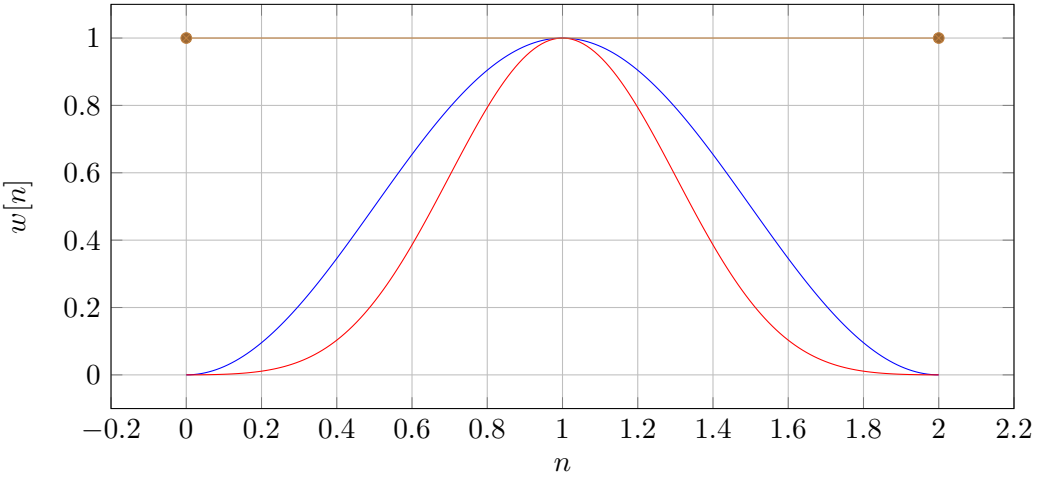


Figure 1.9: Hann, Blackman-Harris and Rectangular window for $N = 2$

Finally the STFT over the sequence (x_N) can be defined as

$$\mathrm{STFT}\{(x_N)\}[\eta, \omega] := \sum_{n=0}^{N-1} x[n] \cdot w[n - \eta] \cdot e^{-i \cdot \omega \cdot n/N}, \quad \eta \in \mathbb{N}, \quad \omega \in \mathbb{R}_+$$

where η stands for the time and ω is the angular frequency which is dependent as $\omega = 2\pi f$ to the frequency f which is analysed. By only using natural numbers in a specific distance for η the `hop_size` of the STFT is introduced in the context of $\eta \in \{k \cdot \texttt{hop\_size} \mid k \in \mathbb{N}\}$. The STFT is a complex-valued function. Taken Euler's formula[5] it can be seen that the imaginary part corresponds to a frequency shifted by a phase of $-\frac{\pi}{2}$. Since humans' perception does not recognise the phase of a wave this information is omitted by taking the absolute value of the STFT. Plotting these magnitude (and additionally squared) amplitudes coded as different colours a 2D image results which is called a spectrogram SPEC which is formally is defined as

$$\mathrm{SPEC}\{(x_N)\}[\eta, \omega] := |\mathrm{STFT}\{(x_N)\}[\eta, \omega]|^2 \quad .$$

In the spectrogram in fig. 1.10 brighter pixels represent the frequencies which occur the most at this point of time.

[4] $a_0 := 0.35875$, $a_1 := 0.48829$, $a_2 := 0.14128$ and $a_4 := 0.01168$
[5] $e^{ix} = \cos(x) + i \cdot \sin(x)$

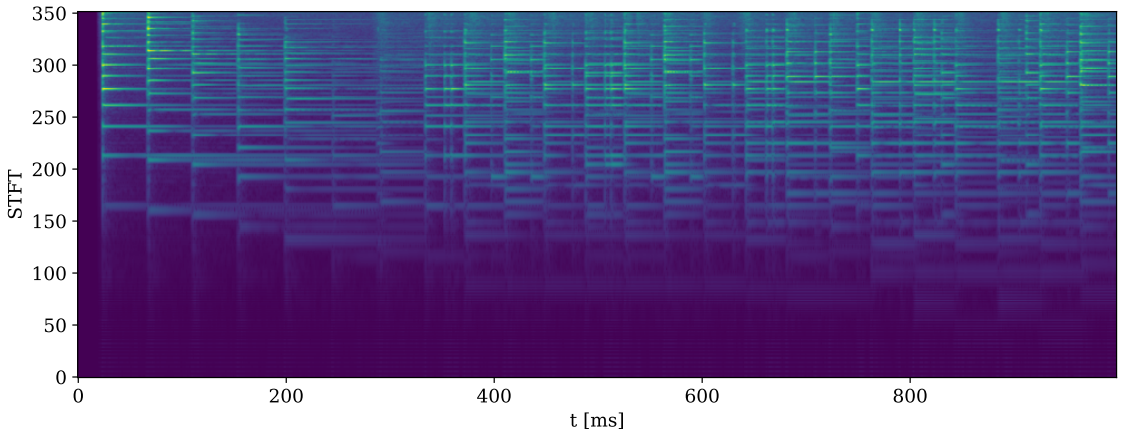


Figure 1.10: Exemplary spectrogram

Another feature which can be extracted from wave mixtures resp. audio signals is the cepstrum. For the calculation on a first step the inverse fourier transformation is applied on the logarithmised spectrogram SPEC. And finally as for the spectrum the magnitude and square of the result is taken.

The spectrogram as well as the cepstrum can be further preprocessed by applying filters. A example for a linear filter is a high-pass or low-pass filter where only high resp. low frequencies are passed. These filters are used to ignore frequencies which are not in the range of the analisis interest.

Based on the the perception of the pitch of a specific frequency the *mel scale* has been introduced. In this logarithmic scale the frequency $f = 1\,\mathrm{kHz}$ correspond to $Z = 1000\,\mathrm{mel}$ and

$$Z = 2595 \cdot \log_{10}\left(1 + \frac{f}{700}\right). \tag{1.10}$$

Using this scale a *triangular filterbank* with mel spaced filters can be applied to reduce the frequency bins. Each filter in the filterbank is shifted equidistantly corresponding to the mel scale. To get the value of a filter as implied by the name a triangular function is applied to the signal which should be preprocessed.

1.6 Sequence to sequence learning

Sequence to sequence (S2S) learning is understood as learning a mapping from an source sequences to a target sequences. In fig. 1.11 the exemplary mapping of the source sequence $[A, B, C, D]$ to the target sequence $[1, 2, 3]$ is visualised. Furthermore it can be seen that a source as well as a target sequence can have arbitrary length.

In 2014 Sutskever et. al [SVL14] introduced a method which transferred this technique to Neuronal Networks based on the Long Short-Term Memory (LSTM). In the following this Neural Network is described.

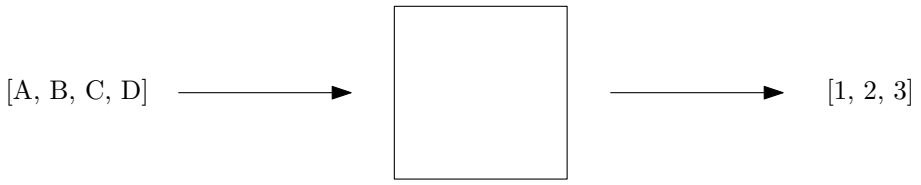


Figure 1.11: Sequence to sequence learning – learning a mapping

1.6.1 LSTM based S2S learning

A LSTM is a recurrent neural network (RNN) which means that the calculated output at time t is again feeded in the network to contribute to the result of the calculation at time $t + 1$.

Coming to Stutskever et. als [SVL14] idea a multilayered LSTM is used to encode the source sequence to a fixed length internal representation. This process of encoding is done until an symbol indicating the end of a sequence (EOS) is propagated. Then a second multilayered LSTM is used to incrementally decode this representation and emit the target sequence. Whereby the emitted token is used as input to calculate the next token of the target sequence. The propagation ends if an (EOS) token leaves the decoder or if a maximal length of the target sequence is reached.

In fig. 1.12 three layers of a LSTM are used as an encoder and a decoder. The same sequences as in the abstracted visualisation in fig. 1.11 are taken. In the first time step the token (A) is propagated through LSTM_1, LSTM_2 and LSTM_3. Just as the (EOS) token appears the vector of fixed dimensionality after the encoder stack results as an embedding for the source sequence. Subsequently the decoding process starts which means that in the first decoding step the decoder stack (LSTM_4 to LSTM_6) consuming the input embedding is emitting the first target token. Iteratively each outputted token (e. g. (1) and (2)) is taken as input for the decoder stack until the (EOS) appears. For the input as well as for the target sequence this token in fig. 1.12 is represented by (]).

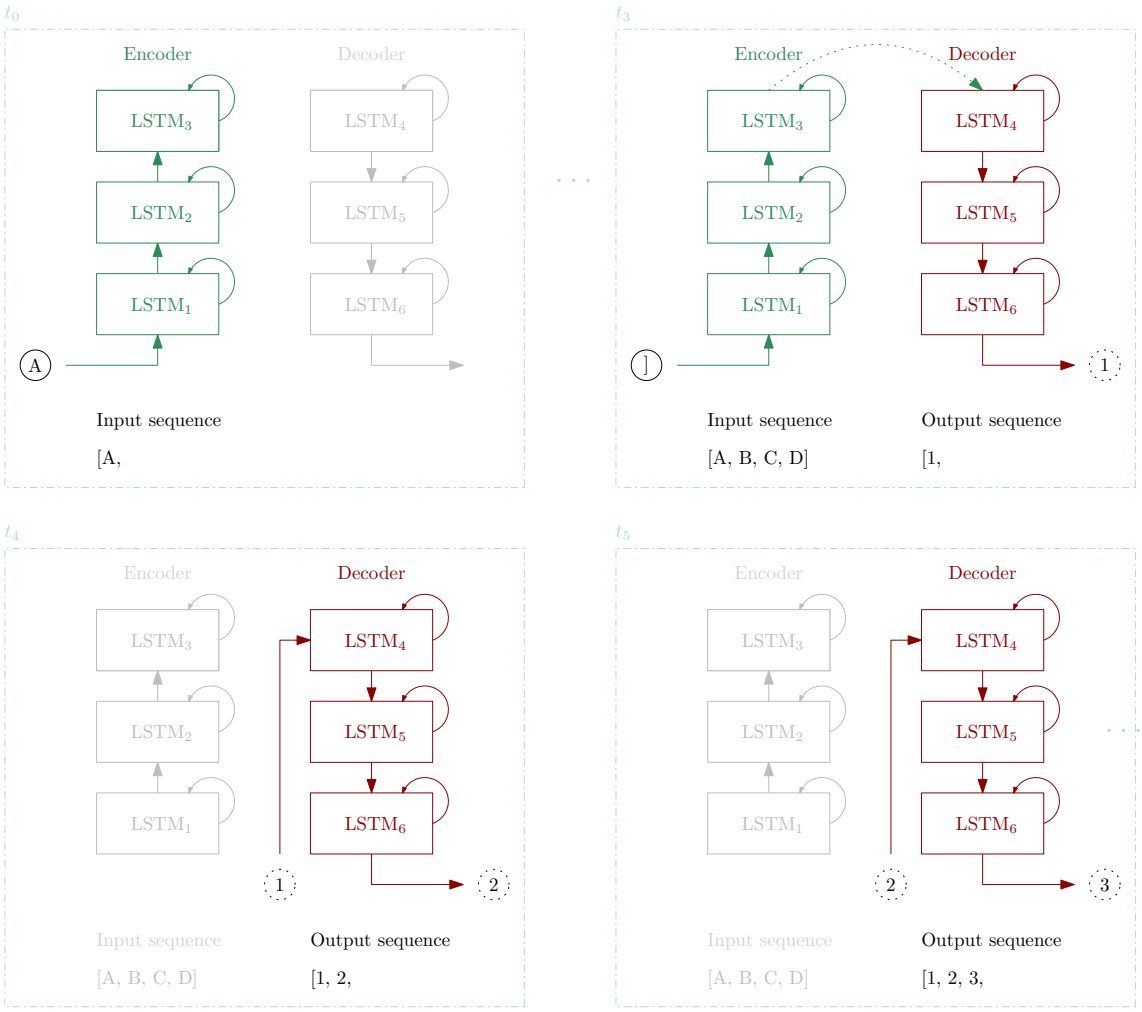


Figure 1.12: Step by step propagation of an Encoder-Decoder LSTM stack

2. Related work

One of the first work about AMT has been published in 1977 by Moorer [Moo77]. As an input two pieces of a synthesised violin resp. guitar duet have been used. The pieces were synthesised to prevent recording noise.

From that year on different publications in this field have been made. In the following recent approaches for music transcription are summarised in a structured way. Furthermore links to other related tasks in a different research field are drawn.

2.1 Music transcription

Recently music transcription is mainly realised by data-driven approaches. Which means that the underlying data is analysed instead of musical knowledge is used to design an algorithm. This data driven algorithms can be divided in two families namely non-negative matrix factorization (NMF) and neural networks (NN). In the following the NMF approach and extensions of it (section 2.1.1) and different NN based approaches (section 2.1.2) are introduced.

2.1.1 Non-negative matrix factorization

The NMF is a set of mathematical techniques where a positive matrix $\mathbf{X} \in \mathbb{R}_{\geq 0}^{m \times n}$ is approximately factorized into two also positive matrices $\mathbf{W} \in \mathbb{R}_{\geq 0}^{m \times r}$ and $\mathbf{H} \in \mathbb{R}_{\geq 0}^{r \times n}$ with $r \leq m$ and $m, n, r \in \mathbb{N}_{>0}$. The approximation is evaluated by the error of reconstruction which is calculated by a const function C defined as

$$C = \|\mathbf{X} - \mathbf{W} \cdot \mathbf{H}\| \quad , \tag{2.1}$$

where $\|\cdot\|$ is often a norm based on the Frobenius norm or on the Kulback-Leibler divergence.

Taking a spectrogram as $\mathbf{X}$ as by Smaragdis et Brown [SB03] a exemplary NMF for $r = 2$ results in a decomposition describing the timely change of frequency bins $\mathbf{H}$ and a dictionary of occurring frequencies $\mathbf{W}$ (c. f. fig. 2.1). In this publication isolated as well

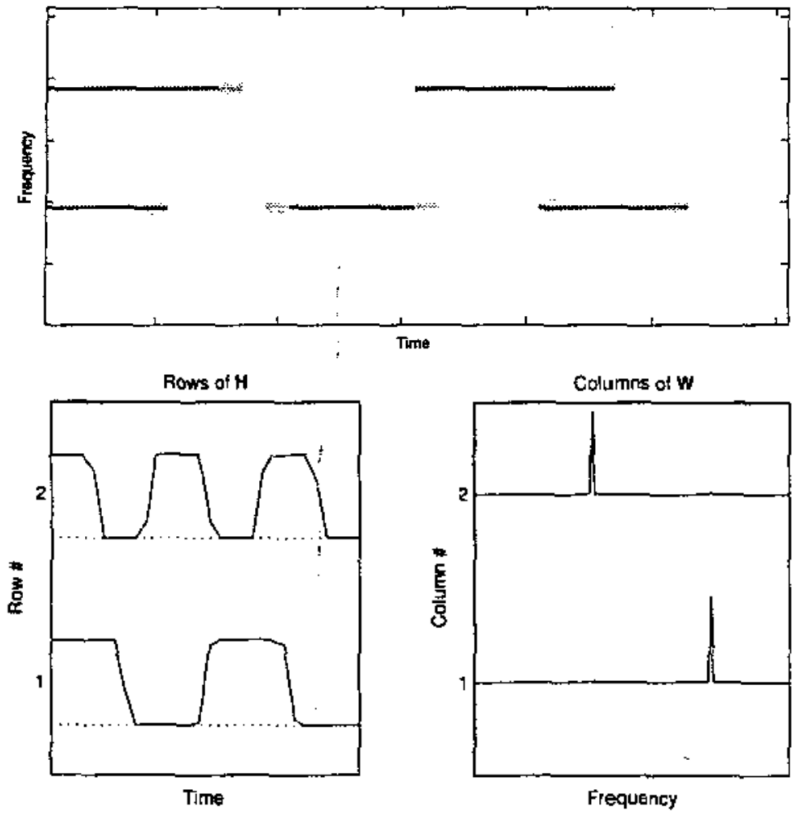


Figure 2.1: NMF for AMT by Smaragdis et Brown [SB03]

as coinciding notes are analysed. The isolate notes have been introduced as monophonic music and the coincident notes as polyphonic in section 1.4.

The NMF can be extended differently: Abdallah and Pumbley [AP06] for example introduced an unsupervised NMF approach with an additional restriction for $\mathbf{H}$ to be sparse[1]. Which as described results in cleaner transcription results for noisy polyphonic recordings. Considering supervised NMF the dictionary $\mathbf{W}$ is estimated by previously analysing additional training material containing e. g. recordings of single notes. Since this results in a overfitting to note dictionary limited to a single instruments Vincent et. al [VBB09] introduced a sub-dictionary representation. In their article a dictionary itself is built as a weighted combination of sub-dictionaries which are representing narrow-bands.

2.1.2 Neural networks

One of the first NN based AMT systems [Mar04] uses Waibel et al.'s time-delay neural networks (TDNN) [WHH+89] to recognise notes. The whole system consists of five modules namely a onset detection, a tracking of partials, a note recognition, a detection of repeated note lengths as well as a loudness estimation. As an feature they use a time-frequency representation preprocessed by 200 filters and furthermore logarithmically spaced.

[1] If more than 50 % of the values of a matrix are zero it is considered to be sparse.

Also recent NN based systems rely on time-frequency representations as a feature. An overview of the information contained in the features, the number of used filters of an applied filterbank and the number of music pieces involved in the training can be examined in table 2.1.

	Feature	# Filters	# Music pieces
[Mar04]	Frequency-representation	200	5
[HES+17]	Spectrogram	229	238
[WCS19b]	Spectrogram and Cepstrum	352	238 resp. 330

Table 2.1: Coarse comparison of different NN based approaches

On of the recent models has been introduced by Hawthorne et al. [HES+17]. As it can be seen in fig. 2.2 two stacks of layers are involved in their onset and frames architecture. In a first step the right part of the architecture is detecting note onset events of a frame. This information is then in a second step used in the left stack to predict the notes per frame. The prediction result is done by a final fully connected sigmoid layer with 88 outputs corresponding to the number of piano keys. As a RNN both stack contain multi-layered Bi-LSTMs.

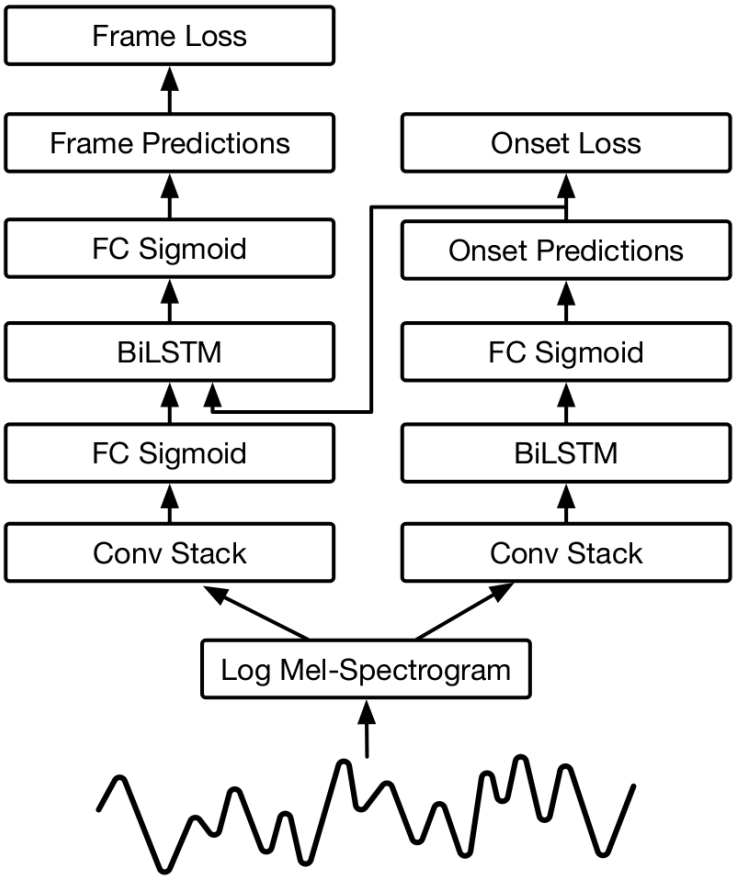


Figure 2.2: Onsets and frames [HES+17] architecture

As an input for their deep convolutional and recurrent network they use a log mel scaled spectrogram. This feature is calculated on 16 kHz audio samples with a FFT window of 2048 and a hop length of 512. Furthermore the data is preprocessed by applying 229 logarithmically-spaced filters and a mel-scale.

In fig. 2.3 an exemplary plotted feature can be examined in the first row. The second row displays the most probable notes calculated by the lower part of the left stack as well as the next row displays probable note onsets events (resp. the output of the right stack). In the last row the fused estimated onsets of the overall network are shown.

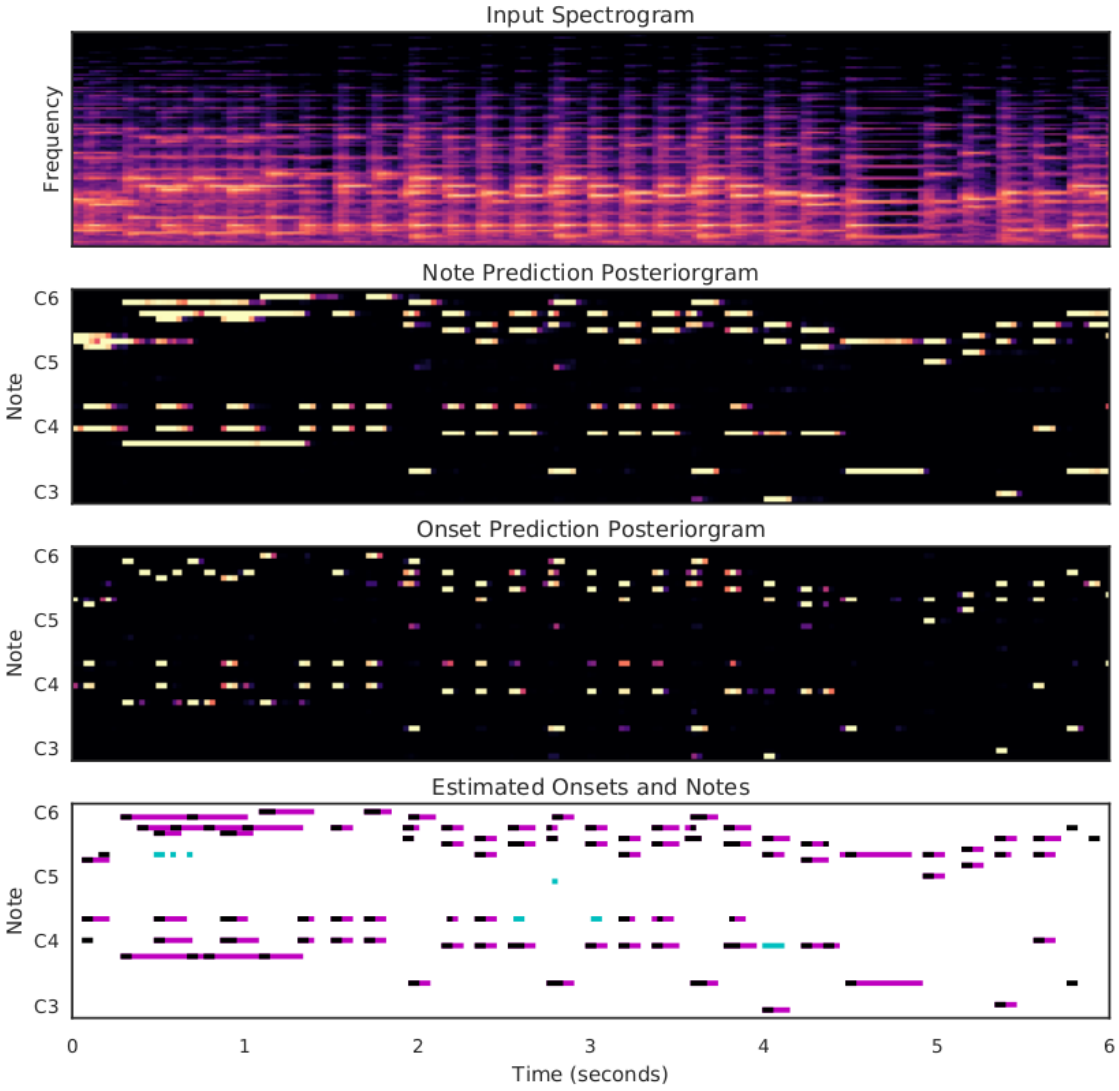


Figure 2.3: Feature and evaluation of the two parts from [HES+17]

As training dataset synthesised as well as real piano recordings are used from the MAPS dataset which will be later introduced in section 2.2.2. A recording of a music piece is cut into 20 s parts to be propagated into the network. Furthermore they also try to cut at points where a small number of notes are active.

The current state of the art paper [WCS19b] is based on the fully convolutional DeepLabV3 network [CPSA17] which is used for semantic segmentation[2]. In semantic segmentation the width and height of an input image equal to the output width and height of a segmentation network. Only a third dimension is added as an output depth where each layer represents a class and a single pixel the possibility a class label assignment. For the AMT task the width of an input image corresponds to the time axis T of a spectrogram and the height represents the frequency bins F as depicted on the left in fig. 2.4. The identical output width and height straightly represents a piano roll with the played notes where the third dimesion represents the different note classes. Formally one pixel $p[t, f]$ represents the probability of a specific note f occurring at a specific point of time t. In addition to it the output layer has been extended by an additional dimension $N+1$ representing N different instrument classes as well as silence. To overcome the class imbalance problem caused by an overrepresented silence an adopted focal loss [LGG+17] is used as a loss function.

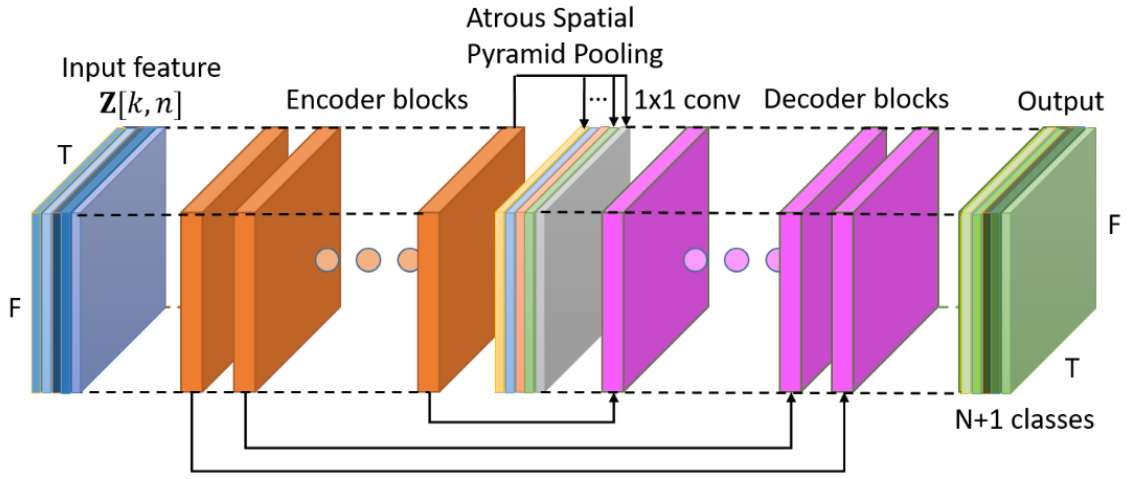


Figure 2.4: AMT using sematic segmentation [WCS19b]

The encoder-decoder architecture of the DeepLabV3 network is fully convolutional[3]. The encoder is connected with the decoder by a Atrous Spatial Pyramid Pooling (ASPP) which calculates scale-invariant features without a loss of resolution.

The aforementioned broadly referred spectrogram in detail is a more complex feature. The representation is introduced as Z_{CFP} resp. Z_{HCFP} which both are multi-channel features. The idea of this feature has been introduced by Wu et. al [WCS18] and is described as CFP (combined frequency and periodicity). The Z_{CFP} gernerally speaking contains one channel of a processed magnitude of the STFT as well as one channel of a preprocessed cepstrum. The preprocessing consists of a high-pass filtering $\mathcal{F}_H$, power scaling by a nonlinear activation function $|\cdot|^\gamma$ and a triangular filterbank $\mathcal{F}_{352}$ with 352 filters[4]. Taking

[2] *Semantic segmentation* describes a neural segmentation algorithm where a class label is assigned to each pixel of an image. For example for autonomous driving regions in camera images are pixelwise labeled as a car, a street or a pedestrian

[3] In contrast to the encoder and decoder layer of the transformer model.

[4] The filters are ranging from 27.5 Hz (A0) to 4,487.0 Hz (C8).

this definitions the feature is calculated by

$$Z_{\text{CFP}} := [Z_{|\text{STFT}|}, Z_{\text{CEP}}] \tag{2.2}$$

$$Z_i := \mathcal{F}^i_{352} \cdot \left|\mathcal{F}^i_H \cdot X_i\right|^{\gamma_i}, \quad i \in \{|\text{STFT}|, \text{CEP}\} \tag{2.3}$$

where the cepstrum X_{CEP} itself is calculated by applying the inverse STFT on $Z_{|\text{STFT}|}$.

Additionally harmonic information is extracted coming to Z_{HCFP}. The idea is to pitch shift the time-frequency representations in a way that harmonic peaks are aligned. This should utilize local convolutional operations to cover harmonic information. $Z_{|\text{STFT}|}$ as well as Z_{CEP} are shifted 11 times which results in a 12 channel representation called Z_{HCFP}.

In fig. 2.5 an exemplary piano roll of the transcription calculated by this paper is shown. The blue lines visualise true positives, green lines false positives and red lines false negatives corresponding to the labels.

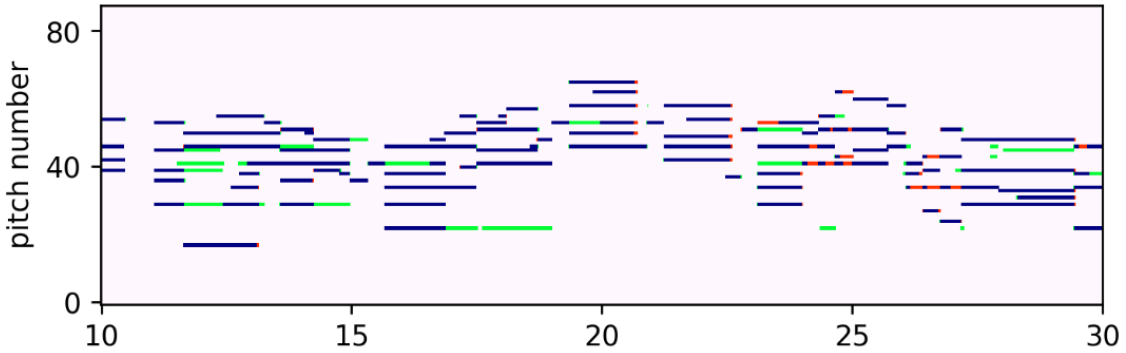


Figure 2.5: Exemplary piano roll evaluation from [WCS19b]

2.2 Datasets

The first publications in MIR were evaluated on small and synthesised datasets. The increase of use of deep neural networks also in music research required a large number of publicly available data which also leads to comparable results. In the following two recently used datasets are presented.

2.2.1 MusicNet

The dataset has been introduced by Thikstun et al. [THK16] in 2016 and contains 34 hours of labled classical music. The set contains 330 freely-licensed music recordings by different chamber music performances. As depicted in table 2.2 the dataset contains music of different instruments and composers. The labels are done by dynamic time warping and a post-verification by trained musicians.

2.2.2 MAPS

The MIDI Aligned Piano Sounds (MAPS) dataset contains 65 hours of piano audio recordings. In total approximately 238 Piano solo pieces of classical and traditional music are

Instrument	Minutes
Piano	1,346
Violin	874
Viola	621
Cello	800
Clarinet	173
Bassoon	102
Horn	132
Oboe	66
Flute	69
String Bass	38
Harpsichord	16

(a) Instruments

Composer	Minutes
Beethoven	1,085
Schubert	253
Brahms	192
Mozart	156
Bach	184
Dvorak	56
Cambini	43
Faure	33
Ravel	27
Haydn	15

(b) Composers

Table 2.2: Statistics about the MusicNet dataset

included. The notes of this music pieces were extracted from MIDI files which are available[5] under Creative Commons license. The dataset has been introduced by Emiya working at Telecom ParisTech in 2008. In table 2.3 groups of the recordings are listed corresponding to the instruments and recording conditions as described by Emiya et. al [EBDB10]. All recordings but `ENSTDkAm` and `ENSTDkAm` which are recorded with a Disklavier are synthesised audio files.

	Instrument resp. software synthesiser	Recording condition
AkPnBcht	Bechstein D 280	Concert hall
AkPnBsdf	Boesendorfer 290 Imperial	Church
AkPnCGdD	Conert Grand D	Studio
AkPnStgb	Steingraeber 130	Jazz club
ENSTDkAm	Disklavier: Yamaha Mark III	Ambient
ENSTDkCl	Disklavier: Yamaha Mark III	Close
SptkBGAm	The Black Grand: Steinway D	Ambient
SptkBGCl	The Black Grand: Steinway D	Close
StbgTGd2	The Grand 2 (Steinberg)	Software default

Table 2.3: MAPS dataset

2.3 Natural Language Processing

The field of Natural Language Processing (NLP) describes how natural language can be analysed and processed by computers. In 2018 Devlin et al. [DCLT18] introduced the language representation model *BERT* which outperformed state-of-the-art systems on eleven different NLP tasks. The tasks range from binary descissions if questions are semantically equivalent (*GLUE MNLI*) to question answering tasks where the network is answering a question by getting content information from a wikipedia article (*SQuAD*). As it can be

[5] Uploaded by Krueger on `http://www.piano-midi.de`.

seen the systems which is based on a S2S neural network called Transformer performed pretty well.

The Transformer has originally been developed by Vaswani et al. [VSP+17] for the machine translation task where sentences sequences from one language are mapped to the translation in a target language. The model out-performed state-of-the-art systems on the WMT 2014 English-to-German as well as English-to-Frensh translation task.

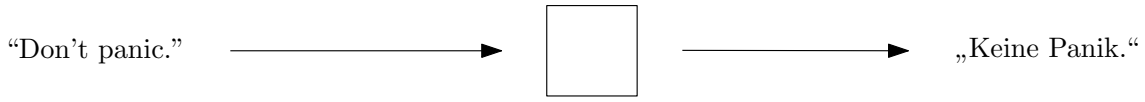


Figure 2.6: Machine translation task

Also in the field of Automatic speech recognition (ASR) the transformer model is used [NSNW19] and is achieving state-of-the-art performance for example for the Switchboard or Fish data. ASR describes the task of analysing spoken text and artificially transcribing it to a text. In fact the ASR task is very similar to AMT where instead of spoken words played notes are used as a source sequence. The target sequence which is a text is replaced by musical notation of the played notes. That similarity conjectures that the transformer

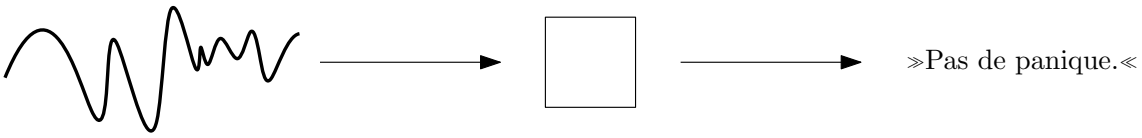


Figure 2.7: Automatic speech recognition task

which already performed well for a variety of tasks could also perform well for the AMT task and is thus introduced in section 3.2.2.

2.4 Music modelling

Another related field of research is music modelling. The aim is to understand musical structure and to artificially rebuild resp. continue music pieces with motifs and phrases. Also in this field the aforementioned transformer has successfully been used by Huan et al. [HVU+18].

In contrary to AMT the task comparable to maschine translation is asking for a mapping between two musical notations resp. textual representations as depicted in fig. 2.8. Furthermore in their work they also deal with a sequential musical textual representation.

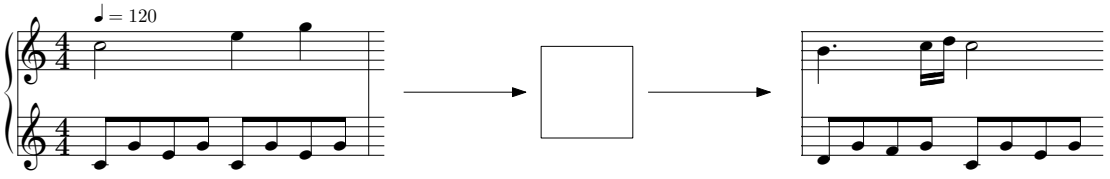


Figure 2.8: Music modelling task

In fig. 2.9 on the top left the initial piano roll of a music piece is shown. Broadly speaking the task for a model is it to continue the initial music piece and in the best case using different structures occurring in the initial piece. The other two plots in the first row are generated by their adapted version of the transformer. And it could be superficially seen that structures of the initial piece are emulated for a longer period of time by the transformer.

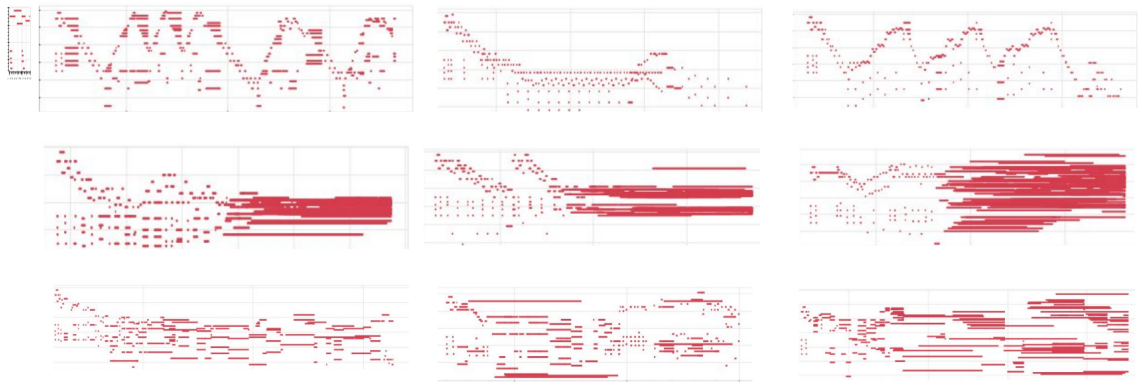


Figure 2.9: Long term structures in music from [HVU+18]

3. Methods

In the following the methods of the thesis are explained. The structure of the chapter can be seen in fig. 3.1. Two used models are introduced in the beginning of this chapter, whereby both models are based on an attention mechanism[1] and thus this mechanism is explained in the very first section. After that the training procedure for the models is explained. In the forth section the features which are propagated through the models are presented. Another essential part of S2S learning are the sequential labels which generation will be explained in a separate section. Also a data augmentation technique and a proof of concept dataset is explained in the end of this chapter.

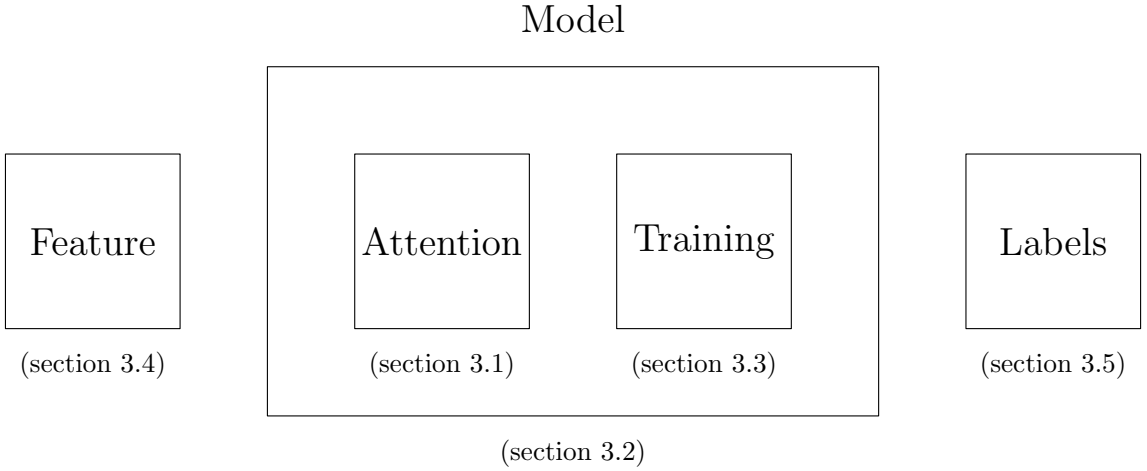


Figure 3.1: Structure of the methods chapter

3.1 Attention mechanism

An attention mechanism is widely used in neural networks in different domains of application, for example for text understanding [HKG+15] or image captioning [XBK+15]. In

[1] The visualisation of the attention and of the transformer are partly inspired by Jay Alammars post which can be found at `http://jalammar.github.io/illustrated-transformer`.

the following the multi-head attention is used. This attention mechanism is more precisely called Multi-Head Scaled Dot-Product Attention and has been introduced by Vaswani et al. [VSP+17] resulting in a neuronal network architecture called Tranformer.

In the following the mechanism is introduced for two input values $\mathbf{v}_1, \mathbf{v}_2 \in \mathbb{R}^{d_k}$. For the attention there exists two additional vectors, first a query vector $\mathbf{q}_1 \in \mathbb{R}^{d_k}$ which can exemplary be interpreted as asking for the first output. Secondly the key vectors $\mathbf{k}_1, \mathbf{k}_2 \in \mathbb{R}^{d_k}$ which are representing each of the input values. As depicted in fig. 3.2 the dot product between the key vectors and the query vector is calculated. The resulting values are scaled by $\frac{1}{\sqrt{d_k}}$ as well as the softmax function is applied to finally take the result and multiply it with the values. The weighted values are then summed to calculate the final representation $\mathbf{o}_1$ of the input for query vector $\mathbf{q}_1$.

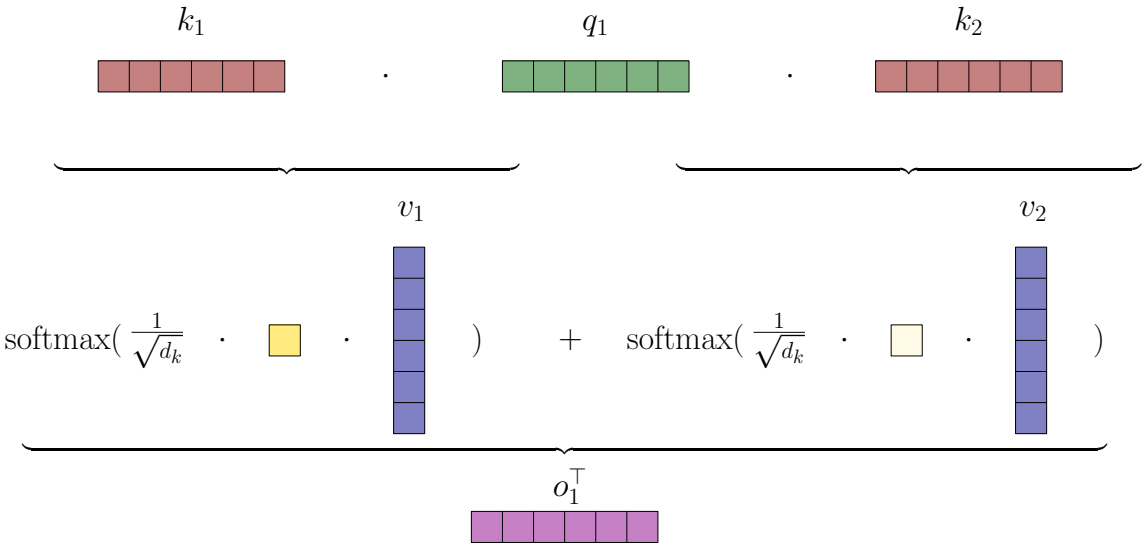


Figure 3.2: Attention for values $\mathbf{v}_i$ calculated by query $\mathbf{q}_i$ and key vectors $\mathbf{k}_i$ for $i \in \{1, 2\}$

Abstracted to more than one query vector and the two key vectors this can be written as a matrix calculation. The different vectors are stacked row-wise to get the matrices Q, K, V as well as resulting in a stacked final representation O as depicted in fig. 3.3.

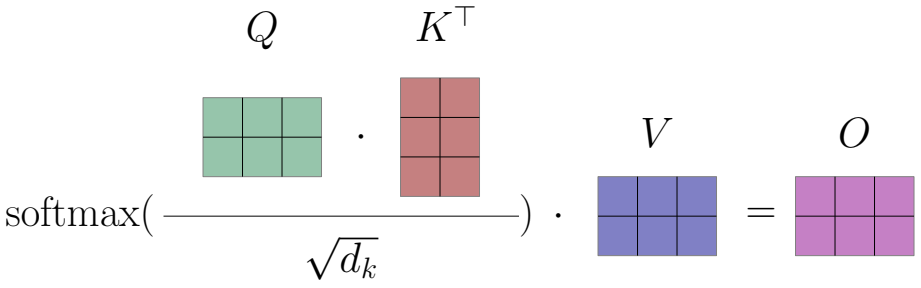


Figure 3.3: Matrix formulation of the Scaled Dot-Product Attention

This mechanism is further extended to not only comprising one triplet of Q, K, V matrices but h, which is also referred as number of heads `n_heads`. All the final representations

calculated by the Multi-Head attention are finally stacked column-wise and fused by a multiplication with a weight matrix W^0.

3.2 Models

For the problem of automatic music transcription two different models for a S2S approach of ASR introduced by Nguyen et al. [NSNW19] are adapted. Namely a model which is based on LSTMs (section 3.2.1) and the already mentioned Transformer (section 3.2.2). Both models are working with the attention mechanism which has been explained in section 3.1. The models are implemented in *Python 3* with the deep learning framework *PyTorch*.

3.2.1 LSTM-based S2S

The model can be structured into an encoder, a decoder and an attention block. As shown in fig. 3.4 the source sequence is propagated through the encoder. The decoder is consuming the already available target sequence which in the beginning will be an empty sequence, because the model has not yet calculated an overall output. As a final step the before mentioned Multi-Head Scaled Dot-Product Attention mechanism Att is applied using the output of the encoder as key k as well as value v and the decoder output as the query q. The tensor calculated by the attention module is finally added to the output of the decoder to calculate the final result. Both blocks contain LSTM resp. BiLSTM layers

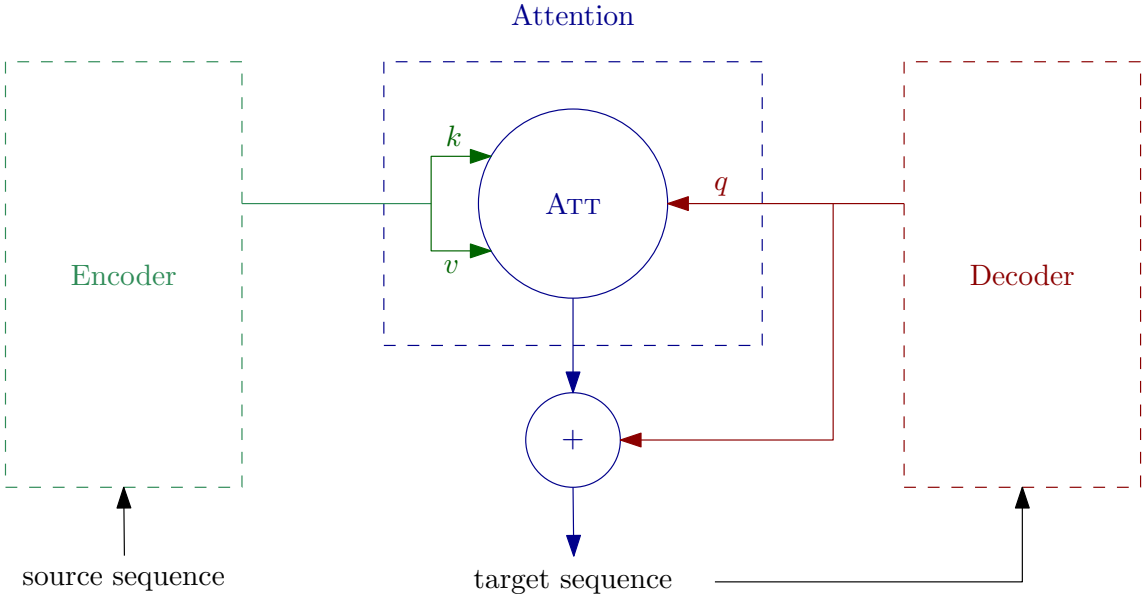


Figure 3.4: Coarse structure of the LSTM-based S2S

which internal features size is controlled by the `d_model` parameter. In the following the two building blocks are explained in detail.

The encoder block consists of an (optional) preprocessing layer (see section 3.2.3) and `n-enc` BiLSTM layers. A BiLSTM written-out Bidirectional LSTM is the combination of two LSTMs where in one of the two LSTMs the sequence is reversely feeded. An output at each

point in time is then concatenatsed shaping the output of the BiLSTM. All but the last BiLSTM layers are as depicted in fig. 3.5 followed by a dropout layer. The dropout rate

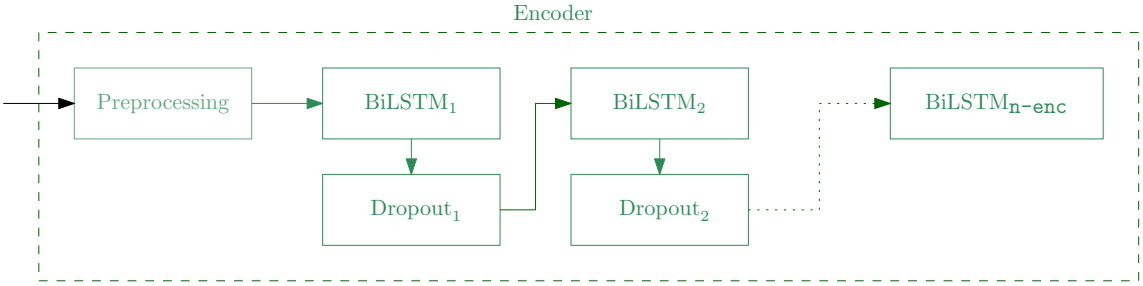


Figure 3.5: Encoder block of the LSTM-based S2S

can be specified by the `dropout` parameter implying the percentage of randomly selected neurons which values are ignored during a propagation.

Another block of the LSTM-based S2S model is the decoder which is shown in fig. 3.6 and consisting of an embedding layer and `n-dec` stacked (unidirectional) LSTM layers. As for the encoder all but the last LSTM layer are followed by a dropout layer. The embedding layer calculates a vectorial representation for each target token and is followed by a embedding dropout layer.

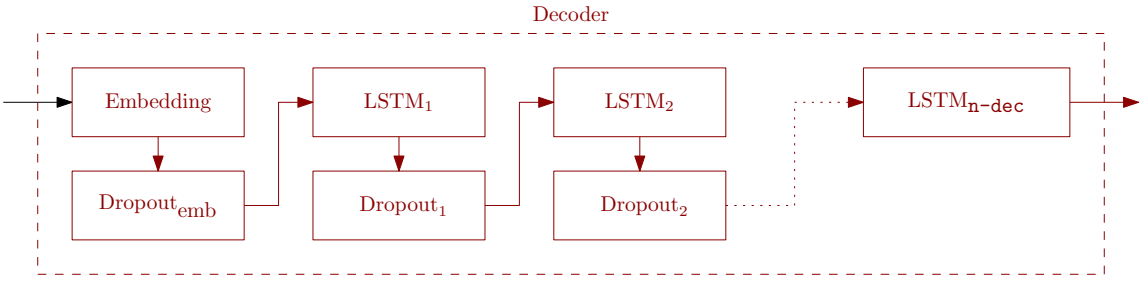


Figure 3.6: Decoder block of the LSTM-based S2S

3.2.2 Transformer

The transformer model as introduced by Pham et al. [PNN+19] for ASR also has an Encoder-Decoder structure (cf. fig. 3.7). The encoder resp. decoder consists of several stacked layers which are introduced in the following.

Since all sequence tokens are propagated simultaneously (in contrary to RNNs) the information of chronological order is lost. Thus the positional embedding layer adds values of a cosine and sinus function with different frequencies to the embedding which uniquely marks each embedded token with the temporal information. In the following pos is describing the position of the token in the sequence and i the position in the embedding dimension of total size `d_model` in the positional embedding PE as

$$\text{PE}(\text{pos}, i) = \begin{cases} \sin\left(\text{pos}/100000^{i/\texttt{d\_model}}\right), & i = 2 \cdot k, \quad k \in \mathbb{N} \\ \cos\left(\text{pos}/100000^{i/\texttt{d\_model}}\right), & \text{else} \end{cases}, \quad \text{pos}, i \in \mathbb{N}_0 \tag{3.1}$$

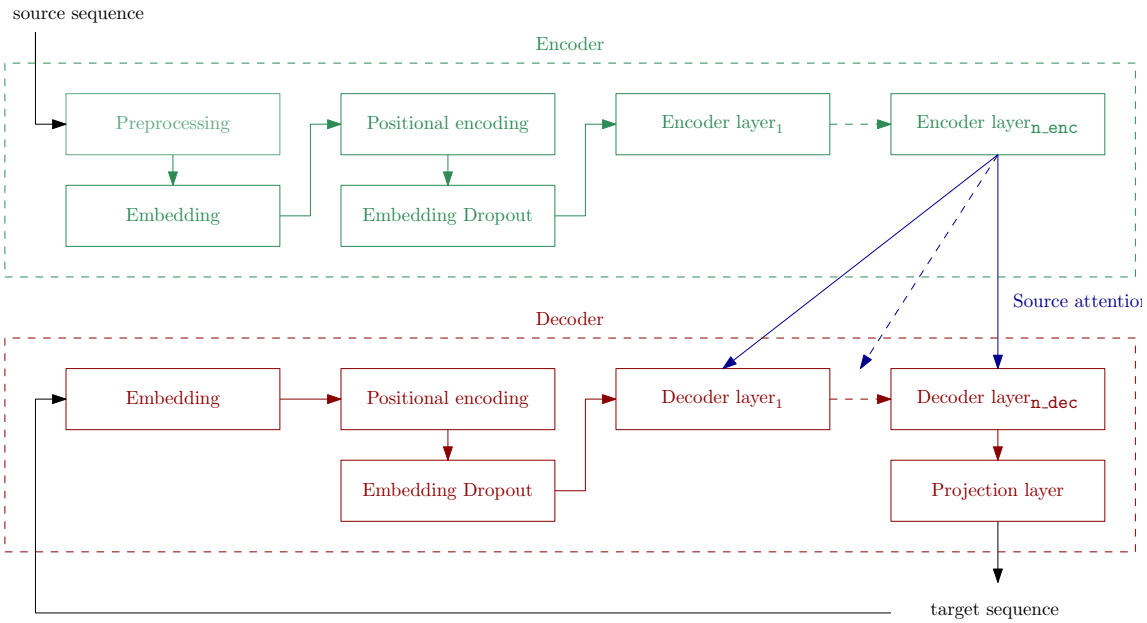


Figure 3.7: Coarse structure of the transformer model

The aforementioned attention mechanism is integrated in three different ways: First as a self attention mechanism in the encoder resp. secondly in the decoder layers. Thirdly a encoder-decoder attention also called source attention which is applied in every decoder layer. In the following the attention mechanism will be explained for all its occurrences.

The stacked layers in the encoder are preceded by an (optional) preprocessing, an embedding, a positional encoding and a embedding dropout layer. One encoder layer is itself a

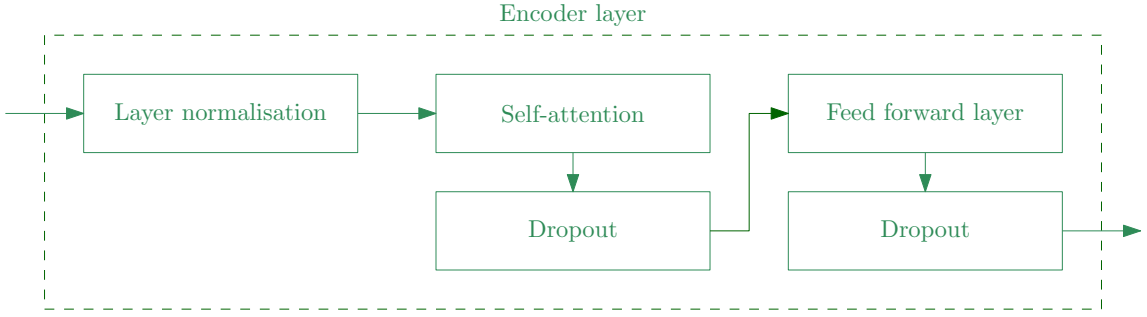


Figure 3.8: Encoder layer of the Transformer

stack of a layer normalization, self-attention, dropout and a feed forward layer. The earlier mentioned self-attention only needs the output of the previous encoder layer resp. for the first layer the positional embedding of the source sequence (both called `enc_output` in the following). The `enc_output` is directly used as a query tensor Q as well as key tensor K. The value tensor V is calculated by an application of a learnable linear layer on `enc_output`.

The modules contained in a decoder layer are the same as in the encoder as you can see in fig. 3.9. The self attention is similar to the attention in the encoder but Q, K and V are

calculated with the previous decoder layer output resp. the result of the embedding in the decoder (both called `dec_output`). But furthermore the decoder additionally contains a source attention. The source attention mechanism allows to decide on which proportion a time step of the encoding is incorporated into the final output. Formally the `dec_output` is used as a query tensor Q and `enc_output` serves as a key tensor K of the attention mechanism Att. As for the self-attention the value vector V is calculated with a linear layer on `dec_output`.

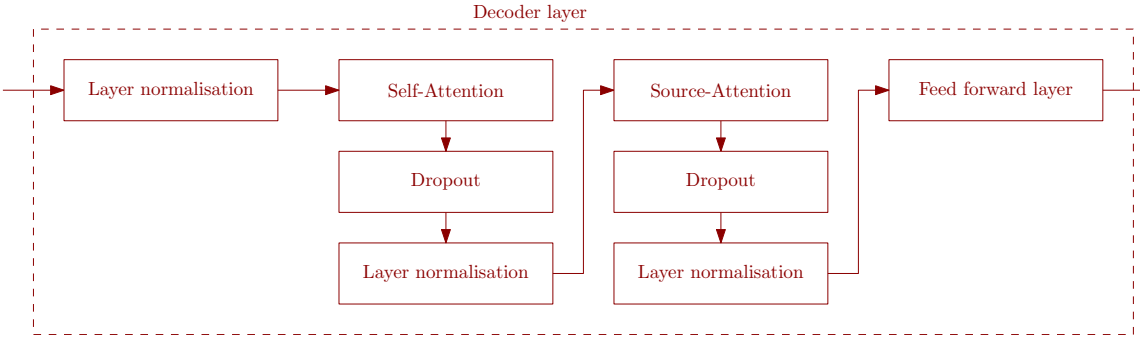


Figure 3.9: Decoder layer of the Transformer

3.2.3 Preprocessing layers

Both introduced models can optionally be augmented by adding a preprocessing layer in front of the embedding layer of the encoder. In the following tree different preprocessing layers have been used: A convolutional layer $\text{Prep}_{\text{conv}}$, an Atrous Spatial Pyramid Pooling layer Prep_{atr} and a convolutional filtering layer Prep_{cf}.

The $\text{Prep}_{\text{conv}}$ is a CNN with two 2D convolution layers which are adapted by the parameter `freq_kn` and `freq_std`. As depicted in fig. 3.10 each of the layers contains 32 filters. The height of a filter kernel is statically set to 3 and the width is adapted by the `freq_kn` parameter. During convolution the kernel is moved 2 pixels in vertical axis and as specified by parameter `freq_std` in horizontal axis.

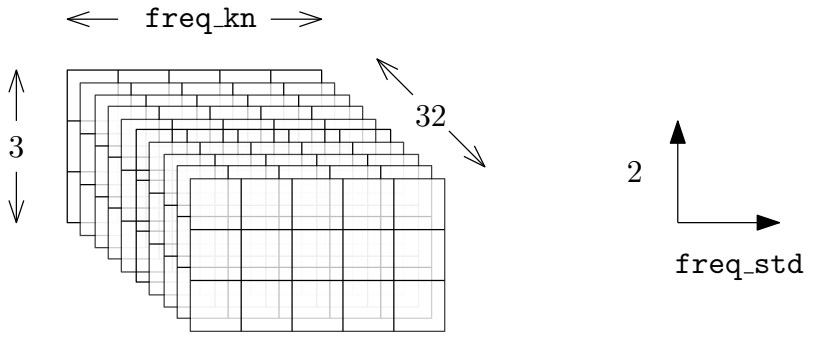


Figure 3.10: CNN kernel used in transformer encoder

The Prep_{atr} is based on Atrous Spatial Pyramid Pooling (ASSP) which has already been used for AMT by Wu et al. [WCS19b]. Originally ASSP has been introduced by Chen et

al. [CPSA17] to extract multi scale feature of a tensor for semantic image segmentation. The PREP_{atr} preprocessing layer is in the here underlying model then further propagated to calculate the source sequence embedding. The convolution used for the feature is called atrous convolution or dilated convolution[2] with a parameter `dilation`. As depicted in fig. 3.11 `dilation` changes the distance between the pixels which are taken for the convolution and multiplied with the kernel. A value of `dilation` := 1 results in a standard convolution. The `dilation` of a dilated convolution increases the receptive field of a filter by using an identical number of parameters.

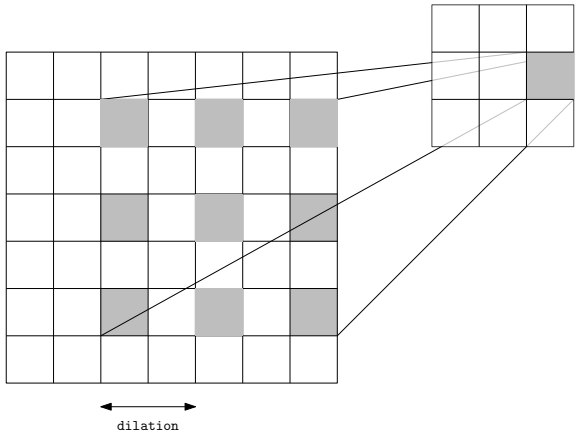


Figure 3.11: Dilated convolution with `dilation` := 2

In contrary to a succession of layers four dilated convolutions and a 2D average pooling are calculated in parallel forming a Spatial Pyramid Pooling. Each of the dilated convolutions uses a different `dilation` parameter to extract feature of different scale. As shown in

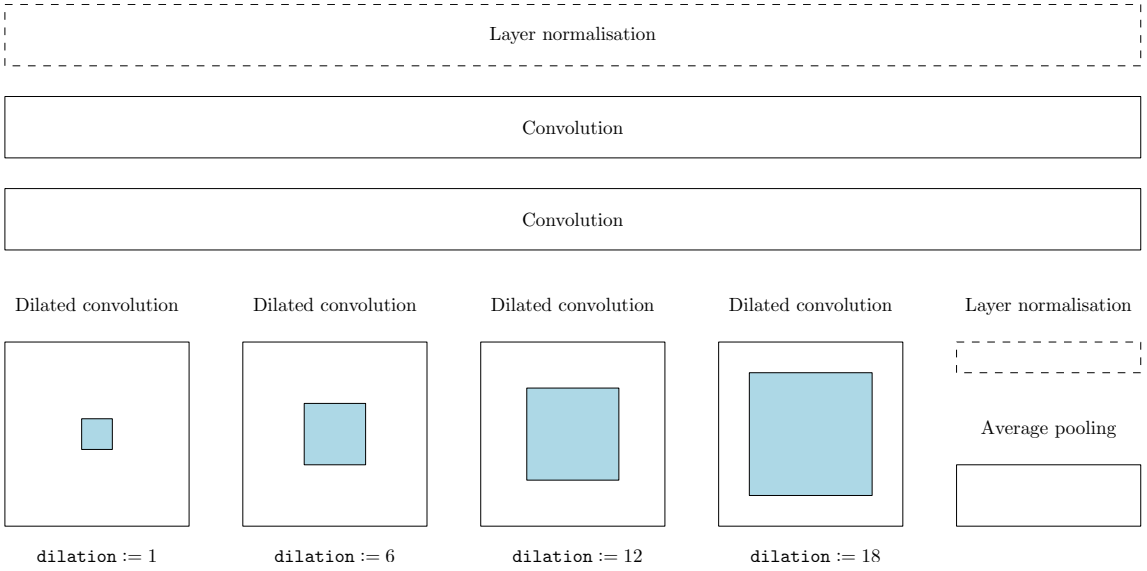


Figure 3.12: PREP_{atr} resp. $\text{PREP}_{\text{atrm}}$ preprocessing layer

[2]Dilated convolution is the name as it has firstly been introduction by Fisher et al. [YK15] in 2015.

fig. 3.12 the `dilation` parameters are $1, 6, 12$ and 18 whereby each convolution is using 16 filters. The thereby extracted 64 channels (16·4) are fused by two successive convolutions resulting in an one channel feature tensor. The dashed boxes in fig. 3.12 symbolise an optional batch normalisation layer which is applied after the average pooling as well as the last fusing 2D convolution. The ASSP preprocessing layer with the batch normalisation is referred as $\text{PREP}_{\text{atrn}}$.

The convolutional filtering PREP_{cf} is expecting unfiltered audio data. This preprocessing layer is replacing the manually applied filter banks which are often applied after the STFT calculation.

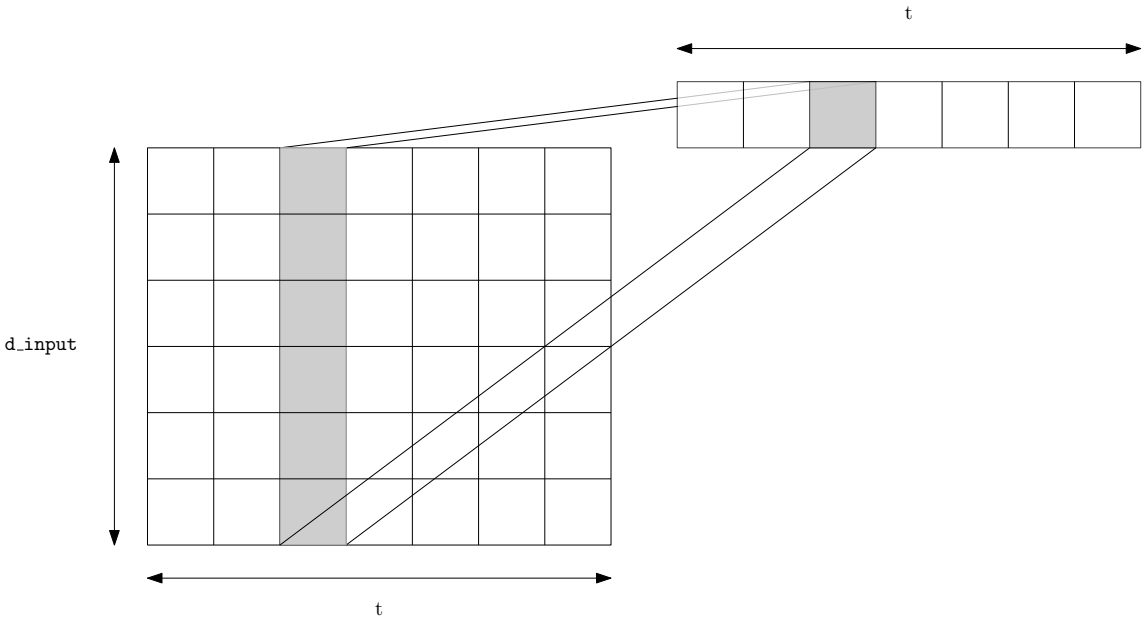


Figure 3.13: Convolutional filtering PREP_{cf}

It is realised by a single convolution layer with a kernel size of $1 \times$ `d_input` (see a exemplary convolution of one filter in fig. 3.13). The parameter `d_input` is the number of frequency values resulting in the STFT. The number of the originally used filters in the filter bank and convolution filters which is 352 is equal. Thus all results of this layers stacked together is resulting (dependent on the learnt kernel values) in a representation comparable to a filter bank calculation.

3.3 Training

For the training the stochastic optimisation algorithm Adaptive Moment Estimation (Adam) is used. The set of parameters at time point t is in the following called $\boldsymbol{\theta}_t$. These parameters are updated as suggested by Kingma et al. [KB14] by

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \frac{\eta}{\sqrt{\hat{v}_t} + \epsilon} \cdot \hat{m}_t \quad .$$

In that formula η is the learning rate, $\hat{m}_t$ the normalised first momentum (mean) and $\hat{v}_t$ the second momentum (uncentered variance). Both momenta are calculated as followed

$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}, \qquad m_t = \beta_1 \cdot m_{t-1} + (1-\beta_1) \cdot g_t,$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}, \qquad v_t = \beta_2 \cdot v_{t-1} + (1-\beta_2) \cdot g_t^2$$

where g_t is the gradient of time step t and α_i, β_i with $i \in \{1, 2\}$ are constant scalars.

Alo a warmum of the learning rate η is applied which means that the learning rate behaves differently in a specific phase at the beginning of the training. More precise η is linearly increasing untill `warmup_steps` steps. This can be formally written as an dependence on t

$$\eta(t) = \texttt{d\_model}^{-0.5} \cdot \min\{t^{-0.5}, t \cdot \texttt{warmup\_steps}^{-1.5}\} \quad .$$

Figure 3.14 is showing the value of η for exemplary values of the constant `warmup_steps`. Experimentally a value of `warmup_steps` := 8.000 gained the best result in the training and is used in the following.

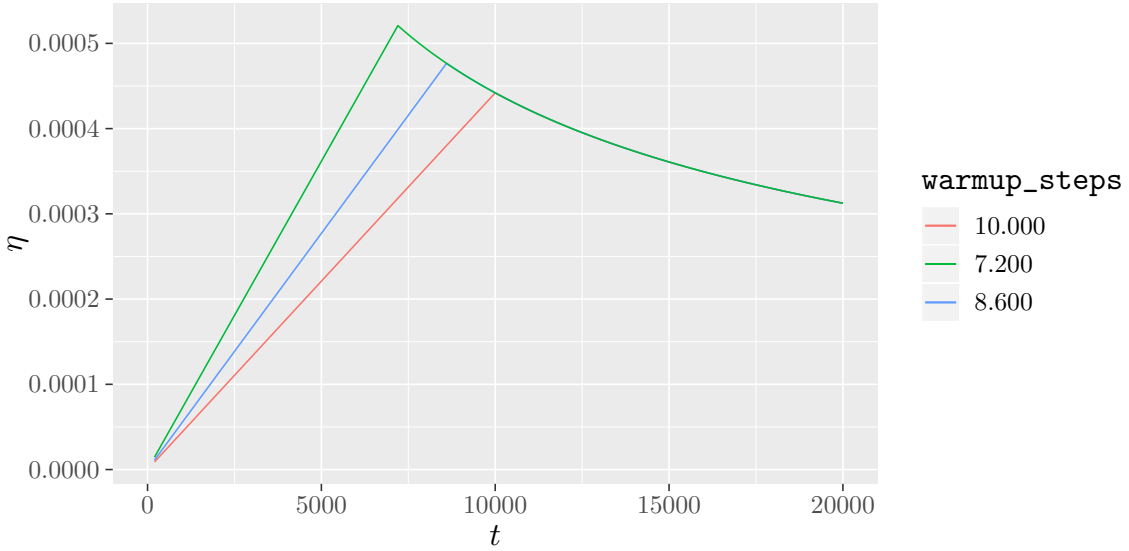


Figure 3.14: Learning rate $\eta(t)$ with `d_model` := 512

3.4 Features

In following four feature are described which are used for the training. The raw audio files of the MAPS and Musicnet dataset are available in a sampling rate of 44.1 kHz. To reduce the amount of data as performed in many publications [WCS19b, ZLL+19, WCS19a, HES+17] the audio files are resampled to 16 kHz. In table 2.1 an overview of the different features which are calculated on the resampled audio files can be examined. Also the corresponding dimension is shown. In the following these features are explained in detail.

The $X_{|\text{STFT}|}$, $Z_{|\text{STFT}|}$ and Z_{CFP} feature is taken from the best performing competitor [WCS19b]. $Z_{|\text{STFT}|}$ are filtered and power scaled STFT amplitudes and Z_{CFP} is a combined

	Feature	Dimension (feature × time)
$X_{\|\mathrm{STFT}\|}$	STFT amplitudes	8000×999
$Z_{\|\mathrm{STFT}\|}$	Filterend and scaled $X_{\|\mathrm{STFT}\|}$	352×999
Z_{CFP}	Cepstrum concatenated with $Z_{\|\mathrm{STFT}\|}$	704×999
Z_{CQT}	Constant-Q transform	253×626

Table 3.1: Features and dimension (for a sample with 20 seconds length)

feature containing STFT information and the cepstrum. In fig. 3.15 an exemplary plot of the combined feature calculated on a 20 seconds MAPS excerpt can be examined. Each column of the plot is representing one step in time. The upper part is the $Z_{|\mathrm{STFT}|}$ feature and the concatenated lower 2D tensor is visualising the cepstrum. For the training of the model with a $\textsc{Prep}_{\mathrm{conv}}$ layer a separate configuration of the dataset of the $Z_{|\mathrm{STFT}|}$ feature without the application of the manual filtering is provided as $X_{|\mathrm{STFT}|}$. For a more precise description of these features see section 2.1.2.

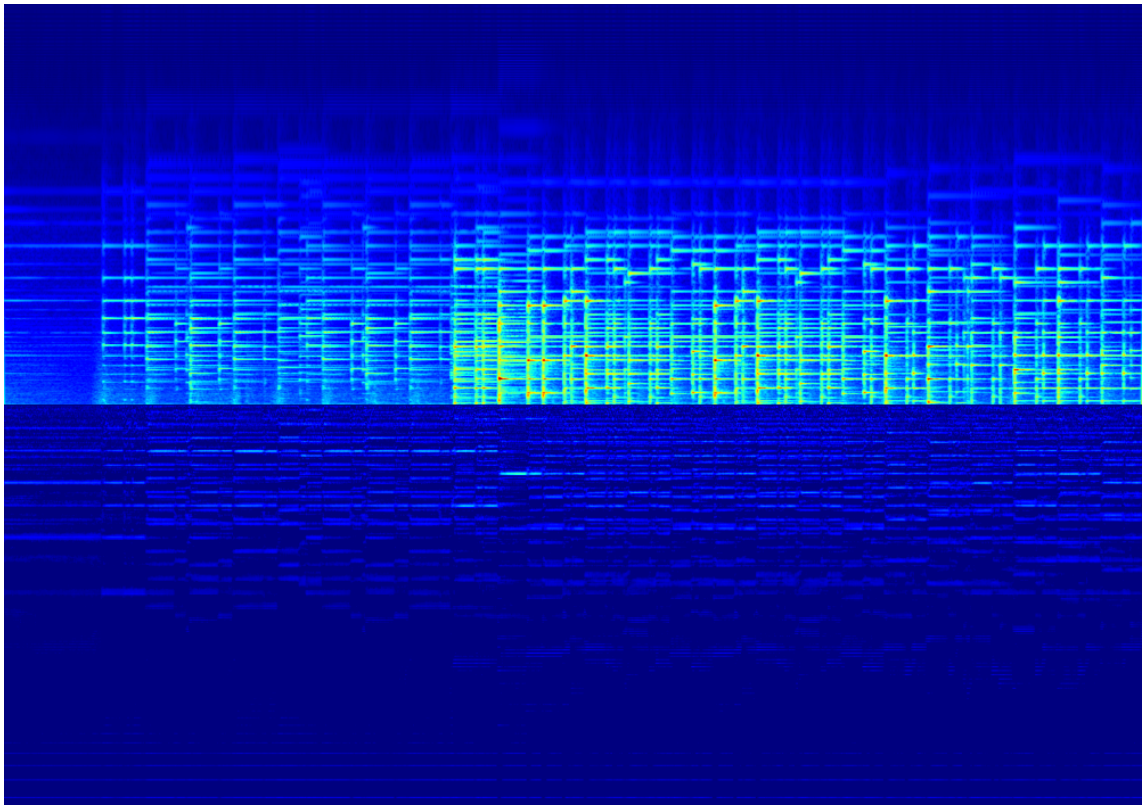


Figure 3.15: Exemplary Z_{CFP} calculated on a 20 seconds MAPS excerpt

Furthermore an own feature $Z_{|\mathrm{CQT}|,\mathrm{Crg}}$ based on the Constant-Q transform (CQT) is introduced. The stacked feature

$$Z_{|\mathrm{CQT}|,\mathrm{Crg}} := [Z_{|\mathrm{CQT}|}, Z_{\mathrm{Crg}}] \tag{3.2}$$

consists of the logarithmically scaled amplitude spectrum of the CQT denoted as $Z_{|\mathrm{CQT}|}$ and a chromagram Z_{Crg}. Similar to the STFT the CQT transform of a discrete time-

domain sequence (c_N) is defined in the following. But instead of ω the analysed frequency is written as $\omega = f_k := f_1 \cdot 2^{k-1/B}$ where f_1 is the frequency of the lowest frequency bin and B the number of bins per octave. Using that and f_s for the sampling rate the transformation can be written as

$$\mathrm{CQT}\{(x_N)\}[\eta, f_k] := \sum_{n=\eta-\lfloor N_k/2 \rfloor}^{\eta+\lfloor N_k/2 \rfloor} x[n] \cdot a_k^*(n - \eta + \frac{N_k}{2}), \quad \eta \in \mathbb{Z},$$

$$a_k(m) := \frac{1}{N_k} \cdot w[\frac{m}{N_k}] \cdot e^{-2\pi i \cdot m \cdot f_k/f_s}, \quad k \in \mathbb{N}_{>0} \quad .$$

It can be seen that compared to the STFT in the CQT the window length is not fixed during the transformation but dependent on N_k. Whereby the value of N_k is dependent on k namely $N_k \propto 1/k$. Regarding the window length this can be interpreted as a higher time resolution as well as a lower frequency resolution for higher frequencies. In the context of musical notes a lower frequency resolution for higher notes is useful in fact the frequency distance of two successive notes is increasing for higher notes as described in section 1.3.

To extract the CQT feature X_{CQT} a Hann window (see section 1.5) is used as w and moved by 512 sample to get successive features. For this feature the CQT uses 229 frequency bins with a minimal frequency $f_{\min} = 32.7\,\mathrm{Hz}$ (C_1) up to 7 octaves. As described above the amplitude value of X_{CQT} is logarithmic scaled to finally get

$$Z_{|\mathrm{CQT}|} = 10 \cdot \log_{10}\left(|X_{\mathrm{CQT}}|^2\right). \tag{3.3}$$

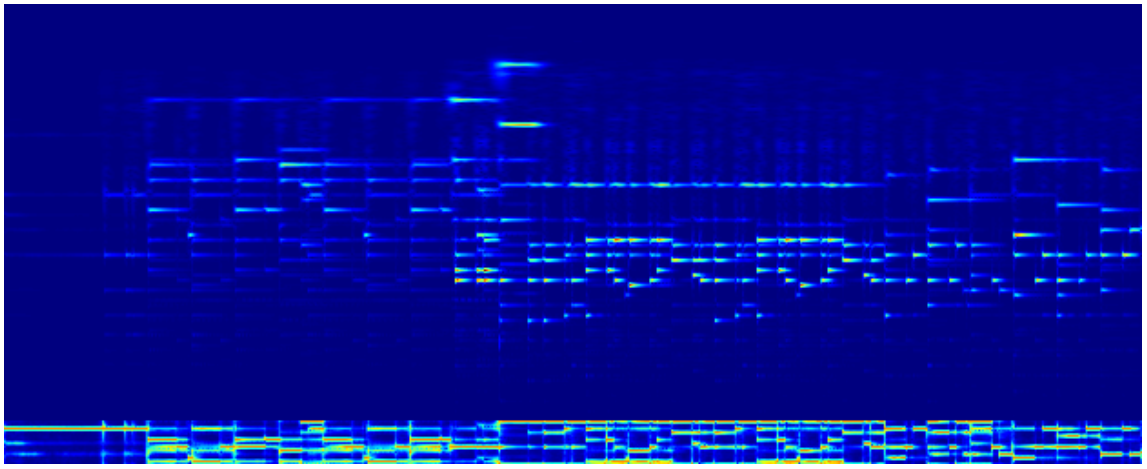


Figure 3.16: Exemplary $Z_{|\mathrm{CQT}|,\mathrm{Crg}}$ calculated on a 20 seconds MAPS excerpt

The chromagram Z_{Crg} is a representation which is related to the pitch classes introduced in section 1.2. Using the idea of Wu et al. [WCS18] of integrating an harmonic information of played music in the feature a chromagram is used. Each of the twelve pitch classes gets two chromagram bins which results in a total number of 24 bins. A Blackmanharris window function (see section 1.5) with window size of 2 is used during the calculation which is performed with the `LibROSA` python library[3]. Figure 3.16 is showing an exemplary plot

[3]More precicely the `librosa.feature.chroma_cqt` function was used to calculate Z_{Crg}.

of the $Z_{|\mathrm{CQT}|,\mathrm{Crg}}$ feature where also each column is representing one time step. The lower part is the 24 pixel sized chromagram Z_{Crg} and the upper part is the logarithmically scaled CQT $Z_{|\mathrm{CQT}|}$.

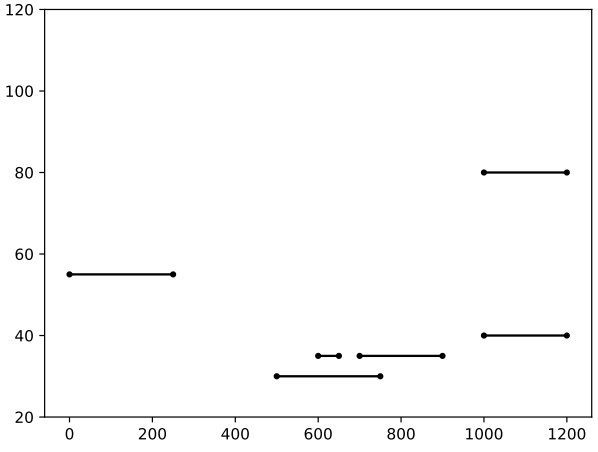
3.5 Labels

In this thesis the MAPS and Musicnet dataset (introduced in section 2.2) are used to train the models. The labels are available as a CSV-file per audio file. Each file contains columns with the labeled note (midi number), onset time and duration (cf. fig. 3.17a). Musicnet also contains an instrument number indicating the instrument playing the note.

The labels can be visualised as a piano roll as in fig. 3.17b.

Note [MIDI]	Onset [ms]	Duration [ms]
55	0	250
30	450	430
35	650	200
35	500	250
40	1000	200
80	1000	200

(a) CSV excerpt



(b) Piano roll

Figure 3.17: Exemplary visualised labels of MAPS resp. Musicnet

3.5.1 Sequencing

Both S2S models expect a sequence as a target mapping and thus the labels need to be processed which can be divided in two steps: On the one hand the segmentation meaning cutting a longer sequence in shorter sub parts. And on the other hand the textual representation which involves the process of finding a mapping between the timely information and a sequential textual representation.

The segmentation is done with two different strategies: One strategy is to alternately cut the audio files into 2000 ms, 2500 ms and 3000 ms parts. The more complex strategy is to cut the files into parts ranging from 2000 ms to 3000 ms by simultaneously minimising the number of fragmented ongoing notes (cf. fig. 3.18). The area shaded in green is searched for a point in time t_p where the number ongoing notes is minimal. If there is more than one possible t_p as in fig. 3.18 the points t_{p_1}, t_{p_2} and t_{p_3} the point absolutely laying closer to 2500 ms which in that case is t_{p_3}. Additionally for both segmentation strategies if

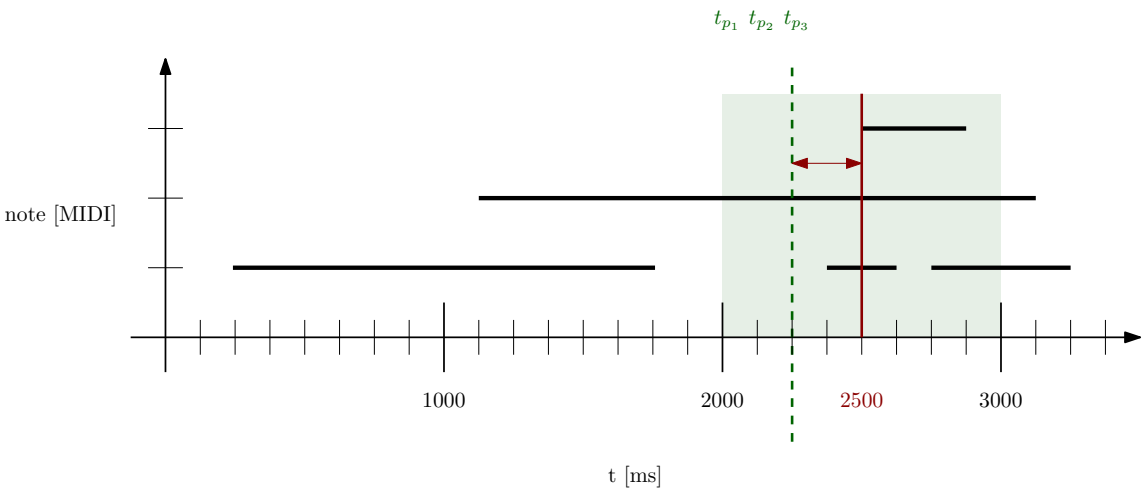


Figure 3.18: Segmentation considering minimal fragmented ongoing notes

the number of labels of a part is exceeding the maximum label length of 500 the part is excluded.

In the following three different textual representations used in this project are introduced: Oore et al. [OSD+18] refer to a representation (in the following called REP$_{\text{Tsh}}$) where a succession of `note-on` and `note-of` tokens is used to describe an onset resp. offset of a musical note. Each MIDI-note has its own `note-on` and `note-of` event resulting in 254 tokens ($= 2 \times 127$). If two notes start or end at the same point in time the onsets are ascendingly sorted corresponding to their MIDI-number. Realising the chronological information after describing the current time step a `time-shift` token is used to start the description of a new point in time. 100 different `time-shift`s describing a shift in time from 10 ms up to 1000 ms are used. This representation (comprising $254 + 100 = 354$ tokens) is used by Magentas music transformer[SO17] which has been mentioned in section 2.4. The sequence matching fig. 3.17 in the REP$_{\text{Tsh}}$ representation can be seen in

```
note_on_55 time_shift_0250 note_off_55 time_shift_0250 note_on_30
time_shift_0100 note_on_35 time_shift_0050 note_off_35
time_shift_0050 note_on_35 time_shift_0050 note_off_30 time_shift_0150
note_off_35 time_shift_0100 note_on_80 note_on_40 time_shift_0200
note_off_80 note_off_40
```

Figure 3.19: Sequence representation REP$_{\text{Tsh}}$ for data from fig. 3.17

fig. 3.19.

The REP$_{\text{Dis}}$ representation is based on Huang et al.s convention [HVU+18] as described for the J. S. Bach chorales dataset. They discretise the time and visually cut horizontal slices from the piano-roll. The notes are then from the top to the botton put together sequentially. Different from the Bach chorales in the underlying datasets of this thesis the number of notes played at a time is not fixed[4]. Thus a `next-time-slice` token in the

[4]For a Bach chorale it is four (called soprano, alto, tenor and bass).

following also symbolised as $\textcircled{\sqcup}$ is introduced indicating the end of a timely slice.

This idea is further extended by a modified subword tokenizer based on the byte-pair encoding (BPE) which has been described by Kudo and Richardson [KR18] as part of the *SentencePiece* neural text processing. The subword segmentation algorithm BPE iteratively fuses two tokens $\textcircled{t_1}$ and $\textcircled{t_2}$ (which could for example be characters) if this specific succession ($\textcircled{t_1}$, $\textcircled{t_2}$) appears the most in all successions in the dataset. In each iteration an additional fused token $\textcircled{t_{(1,2)}}$ is added to the dictionary and replaced at all occurrences in the dataset. Also this new tokens can be multiply grouped together, e. g. $\textcircled{t_{(1,2)}}$ and $\textcircled{t_3}$ to $\textcircled{t_{(1,2,3)}}$. Similarly to the mentioned BPE a succession of tokens in $\textsc{Rep}_{\text{Dis}}$ between two $\textcircled{\sqcup}$ (in the following called time slice $\text{Ts}_{\textcircled{\sqcup}}$) can also be fused. The resulting new tokens can be seen as chord which persists of single notes. Different from words resp. characters the order of tokens of $\text{Ts}_{\textcircled{\sqcup}}$ is arbitrary relating to the resulting sound. Thus tokens of a time slice are mathematically interpreted as a set induced by a union rather than a succession. Hence finding the most common succession is now a search for the most frequent subset in the union. Just as in BPE only two tokens are fused also the subset check is performed only for two sets of the unions of $\text{Ts}_{\textcircled{\sqcup}}$.

In fig. 3.20 BPE is exemplary run on a character and a note sequence. As it can be seen the $\textcircled{\texttt{H}}$ and $\textcircled{\texttt{e}}$ are fused to a $\textcircled{\texttt{He}}$ token. Comparable the sets of notes $\{31\}$ and $\{42\}$ of one $\text{Ts}_{\textcircled{\sqcup}}$ are fused to $\{31\}$. Please note that the written order of the notes in one $\text{Ts}_{\textcircled{\sqcup}}$ is irrelevant and it is difficult for an humans to identify a pair occurring in two $\text{Ts}_{\textcircled{\sqcup}}$ s. In

Character based	**Note based**		
Hello␣world.	$\{20\} \cup \{31\} \cup \{43\} \cup \{31\}$	␣	$\{20\} \cup \{31\} \cup \{2\} \cup \{12\} \cup \{43\}$
$\textcircled{\texttt{He}}$llo␣world.	$\{31, 43\} \cup \{20\} \cup \{31\}$	␣	$\{31, 43\} \cup \{20\} \cup \{2\}$

Figure 3.20: BPE for characters and notes

the following this representation is called $\textsc{Rep}_{\text{Dis-bpe}}$.

3.5.2 Label furnishing

Before the token sequences described above are fed into the models the sequences are scanned and all occurring tokens are added to a dictionary. All tokens which occur at minimum five times are kept for the training while the others are replaced by the unknown token $\textcircled{\texttt{unk}}$. Also later sequences are padded by a $\textcircled{\texttt{pad}}$ token videlicet as many as needed to form batches of same length. Eventually each sequence is prefixed by a beginning of sequence token $\textcircled{\texttt{bos}}$ and suffixed by an end of sequence token $\textcircled{\texttt{eos}}$. These tokens are needed to indicate the model the start resp. end of the propagating sequence.

3.5.3 Training

The extracted features as explained in section 3.4 are stored in several *ark-files*. To efficiently feed the data and labels into the model the parameter `b_input` which adjusts the number of inputs for one batch and needs to be adjusted according to the available storage of the GPU.

Before these *ark-files* files can be fed in the models the labels consisting of a sequence of tokens need to be converted to be used for the loss calculation. Thus the tokens are converted to a vectorial representation using the *one-hot encoding*. As depicted in fig. 3.21

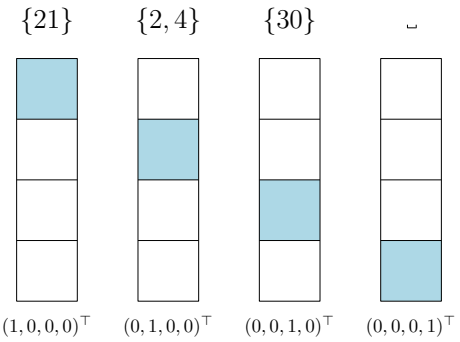


Figure 3.21: Exemplary one-hot encoding for $\text{REP}_{\text{Dis-bpe}}$

where the dictionary contains four entries namely $\boxed{\{21\}}$, $\boxed{\{2,4\}}$, $\boxed{\{30\}}$ and $\textcircled{\sqcup}$ the one-hot vector dimension equals the dictionary size resp. the number of classes `n_classes`.

The output of the models which is calculated by a softmax-layer can also bee seen as such an encoding. To calculate the loss for a propagation the *cross entropy loss* is used. The cross entropy H_{CE} for two discrete probability distributions $\mathbf{p}, \mathbf{q} \in \mathbb{R}^d_{\geq 0}$ is calculated by

$$H_{\text{CE}}(\mathbf{p}, \mathbf{q}) = -\sum_{i=0}^{\texttt{n\_classes}} p_i \log(q_i) \quad .$$

As for the one-hot vectors also `n_classes` specifies the dimension of the output of the last softmax-layer resp. the dimension of the one-hot vetors. For the training task $\mathbf{p}$ describes the target labels and $\mathbf{q}$ the actual prediction of the model.

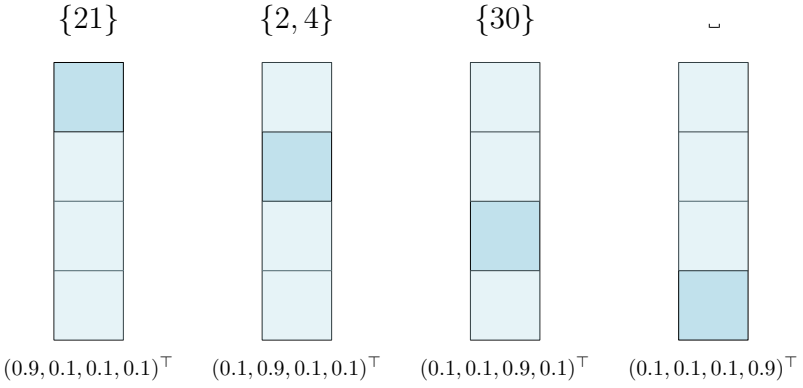


Figure 3.22: Exemplary smoothed ($\epsilon := 0.1$) one-hot encoding for $\text{REP}_{\text{Dis-bpe}}$

Also label smoothing is applied which means that negative target values ($p_i = 0$) are set to a small ϵ and positive target labels ($p_i = 1$) are set to $(1 - \epsilon)$ (c.f. fig. 3.22). In the training of this project the smoothing factor is set to $\epsilon := 0.1$. This strategy is resulting in a *smoother* gradient and has shown its effectiveness in recent publications.

3.6 Data augmentation

To overcome the problem of overfitting several data augmentation techniques exists. For the training SpecAugment which has been introduced by Park et. al [PCZ+19] is used. It was originally applied in the domain of ASR. The spectrograms can be manipulated in three ways: First the time axis of a spectrogram will be warped. Secondly some frequency are masked out as well as thirdly some time steps are dropped.

In the training procedure the two last techniques were applied which are exemplary plotted in fig. 3.23. Two pairs of the $Z_{|\mathrm{STFT}|}$ feature are plotted where on each right plot the random frequency and time step masking is applied. In the implementation used for this training on default settings $n = 2$ drops for the frequency axis as well as for the time axis are applied. The dropped out values are uniformly random selected for each drop. Whereat the maximal dropped time values is set to 20 % and 15 % for the frequencies with default settings.

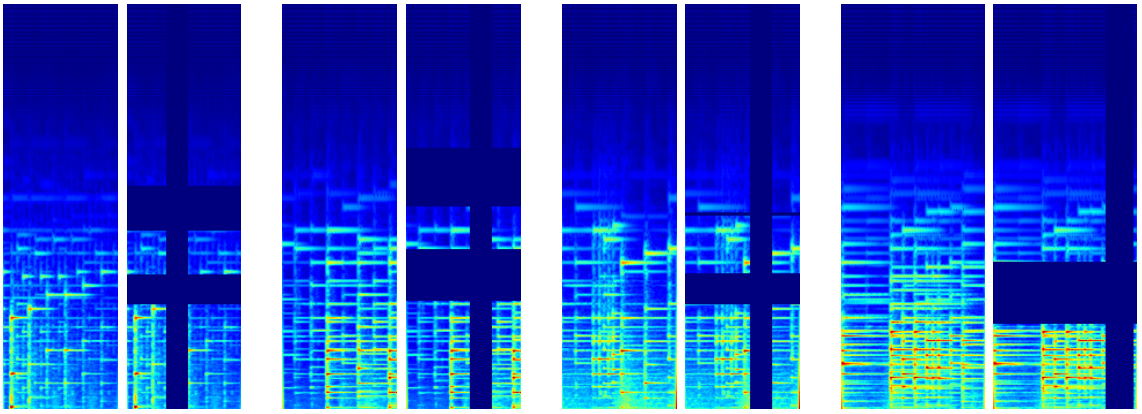


Figure 3.23: Exemplary masked out frequency and time values on $Z_{|\mathrm{STFT}|}$ of MAPS

3.7 Synthetic Bach dataset

As a proof of concept a self created monophonic dataset of 150 bach music pieces has been created (c.f. fig. 3.24). The bach music pieces are shipped with the python library

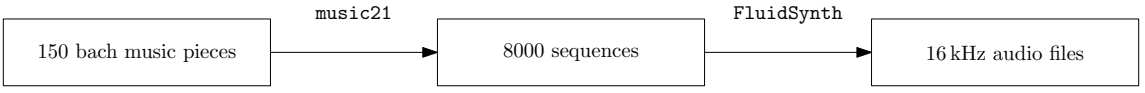


Figure 3.24: Synthetic Bach dataset generation

`music21`[5]. Polyphonic parts of a piece are extracted as separate part books resulting in approximately 8.000 sequences. The open source software synthesiser `FluidSynth`[6] is applied to export the synthetic 16 kHz audio files which can then be used to calculate the above mention features.

[5] `https://web.mit.edu/music21`
[6] `http://www.fluidsynth.org`

For the purpose of testing the whole pipeline the $Z_{|\mathrm{STFT}|}$ features were extracted from the bach sound files. An exemplary plot of the feature and the corresponding labeled notes can be examined in fig. 3.25. Also the harmonics of a note in the $Z_{|\mathrm{STFT}|}$ which results in

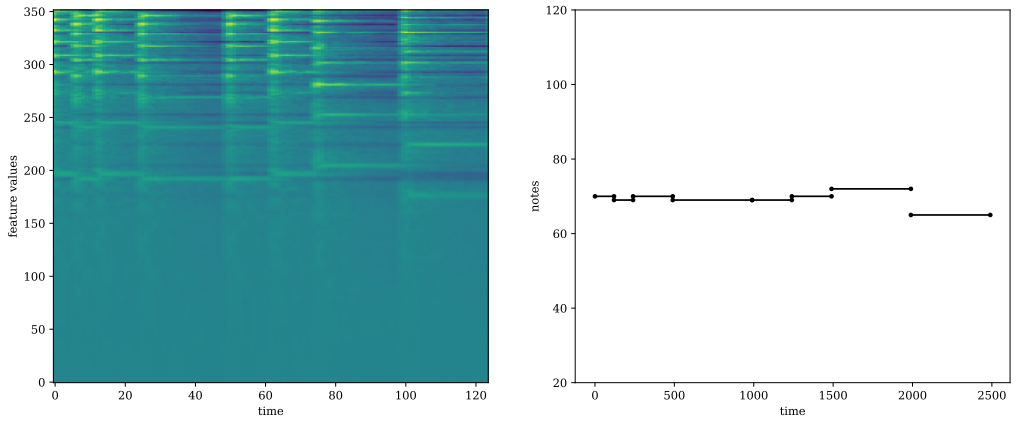


Figure 3.25: Exemplary $Z_{|\mathrm{STFT}|}$ feature of the synthetic bach dataset and labels

more than one active frequency per time step in the plot can be seen.

Furthermore this data is then taken to train a transformer[7] and a LSTM-based S2S[8] model for 40 epochs. As it can be seen in fig. 3.26 the ppl is vastly dropping in both models to a value close to one. The upper plot is showing the ppl on the validation subset and the lower plot the ppl on the trainind subset. The fast drop of the ppl to a value close to one implies that the model is able to learn the mapping from the features to a monophonic note sequence.

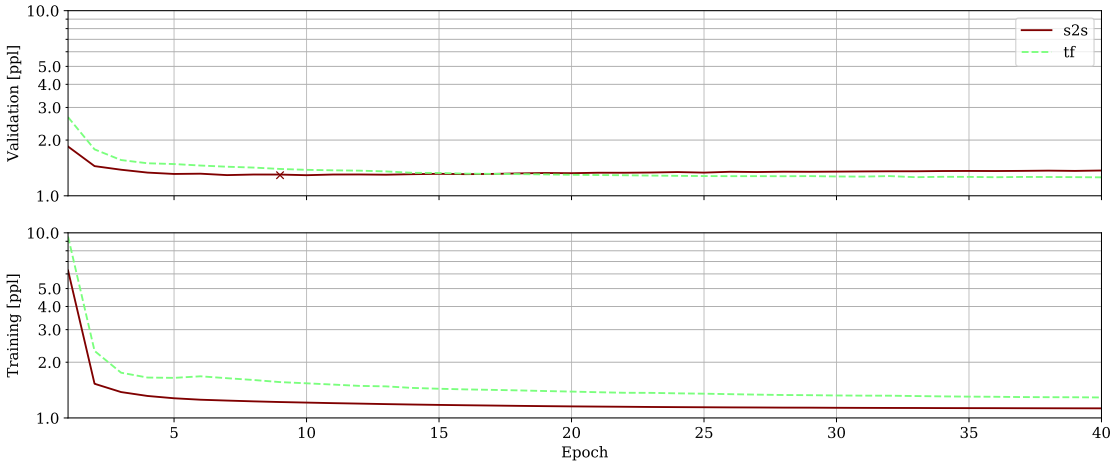


Figure 3.26: Training with the synthetic bach dataset

[7]Tranformer with 4 encoder and decoder layers
[8]LSTM-based S2S model with 4 encoder and decoder layers

4. Evaluation and results

4.1 Experimental settings

For the evaluation the standardised framework for MIR tasks *mir_eval* which has been introduced by Raffel et al. [RMH+14] is used. The appropriate evaluation function from the framework is used to evaluate the results of the *multiple-F0* estimation and tracking task which can also more generally be called automatic music transcription. The token predictions are reinterpreted as a piano roll as well as the target labels. The time axis is then discretised with a 0.02 s resolution which corresponds exactly to the 128 pixel output of Wu et al.'s model [WCS19b] for 2.56 s test samples.

Using the function from `mir_eval.multipitch` with default parameters the number of true positives $\mathrm{TP}^{(s)}$, false negatives $\mathrm{FN}^{(s)}$, false positives $\mathrm{FP}^{(s)}$ and true negatives $\mathrm{TN}^{(s)}$ are calculated per sample s.

With these values precision $\mathcal{P}^{(s)}$, recall $\mathcal{R}^{(s)}$ and thereof the $F_1^{(s)}$-score for each segment s is calculated by

$$\mathcal{P}^{(s)} = \frac{\mathrm{TP}^{(s)}}{\mathrm{TP}^{(s)} + \mathrm{FP}^{(s)}}, \quad \mathcal{R}^{(s)} = \frac{\mathrm{TP}^{(s)}}{\mathrm{TP}^{(s)} + \mathrm{FN}^{(s)}} \quad \text{and} \quad F_1^{(s)} = \frac{2 \cdot \mathcal{P}^{(s)} \cdot \mathcal{R}^{(s)}}{\mathcal{P}^{(s)} + \mathcal{R}^{(s)}} \quad . \tag{4.1}$$

Finally the F_1-score of each segment s is averaged by the arithmetic mean to get the overall score $\overline{F_1}$ of all n segments as well as the precision resp. the recall with

$$\overline{F_1} = \frac{1}{n} \cdot \sum_{s=1}^{n} F_1^{(s)}, \quad \overline{\mathcal{P}} = \frac{1}{n} \cdot \sum_{s=1}^{n} \mathcal{P}^{(s)}, \quad \overline{\mathcal{R}} = \frac{1}{n} \cdot \sum_{s=1}^{n} \mathcal{R}^{(s)} \quad . \tag{4.2}$$

4.1.1 Data

The models are trained on different processed versions of the MusicNet and MAPS dataset which are explained in the following. Both datasets were introduced in section 2.2. For the evaluation of the different versions dedicated evaluation subsets are used for the MAPS dataset which are also explained in the following.

MAPS

As suggested in [WCS19b] the recordings of the real pianos (60 pieces) are hold out for testing resp. calculating the F_1 score of a model. The other synthesised recordings are used for training (180 pieces) and validation (30 pieces). This train-test-validation-split can in detail be seen in table table 4.1 where each row corresponds to 30 music pieces.

	Subset
AkPnBcht	Train
AkPnBsdf	Train
AkPnCGdD	Train
AkPnStgb	Train
ENSTDkAm	Test
ENSTDkCl	Test
SptkBGAm	Train
SptkBGCl	Train
StbgTGd2	Validation

Table 4.1: MAPS dataset train-test-validation split subsets

The model is trained with the train subset and overfitting is checked on the validation subset during training as well as optimisation of other hyperparameters. There exist seven different versions of the train and validation subset which are are calculated by processing the raw 16 kHz audio files and text labels files of MAPS. The described versions are summed up in table 4.2. For six versions the audio files are evenly cut into 2 second, 2.5 second and 3 second ($\{2, 2.5, 3\}$) utterances by the same label segmentation strategy. The feature of three of this versions is $Z_{|\text{STFT}|}$ from [WCS19b]. On one of these versions the labels are calculated by the REP_{Tsh} convention resulting in the version $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$. Similar to this $\texttt{data}_{|\text{STFT}|,\text{Dis}}$ and $\texttt{data}_{|\text{STFT}|,\text{Dis-bpe}}$ corresponds to the REP_{Dis} resp. $\text{REP}_{\text{Dis-bpe}}$ as a textual label representation. In $\texttt{data}_{\text{CFP,Tsh}}$ also the cepstrum is integrated using the feature Z_{CFP}. For the model which has a layer to learn the filtering which is done in preprocessing for example for $Z_{|\text{STFT}|}$ a version without the filtering in preprocessing $\texttt{data}_{|\text{stft}|,\text{Tsh}}$ is provided. Another version $\texttt{data}_{|CQT|,\text{Tsh}}$ is based on the self crafted feature $Z_{|CQT|}$. And the seventh version $\texttt{data}_{|\text{STFT}|,\text{Tsh,min}}$ is using another label segmentation strategy where the number of ongoing notes on a cut is minimised.

	Feature	Segmentation	Textual representation	# utterances
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$	$Z_{\|\text{STFT}\|}$	$\{2, 2.5, 3\}$	REP_{Tsh}	16.429
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$	$Z_{\|\text{STFT}\|}$	$\{2, 2.5, 3\}$	REP_{Dis}	15.622
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$	$Z_{\|\text{STFT}\|}$	$\{2, 2.5, 3\}$	$\text{REP}_{\text{Dis-bpe}}$	15.622
$\texttt{data}_{\text{CFP,Tsh}}$	Z_{CFP}	$\{2, 2.5, 3\}$	REP_{Tsh}	16.429
$\texttt{data}_{\|\text{stft}\|,\text{Tsh}}$	$X_{\|\text{STFT}\|}$	$\{2, 2.5, 3\}$	REP_{Tsh}	16.429
$\texttt{data}_{\|CQT\|,\text{Tsh}}$	$Z_{\|CQT\|}$	$\{2, 2.5, 3\}$	REP_{Tsh}	16.429
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh,min}}$	$Z_{\|\text{STFT}\|}$	min	REP_{Tsh}	16.554

Table 4.2: MAPS dataset versions

For the final evaluation the test set is split into fixed sized utterances of 2.56 seconds which results in 6.111 utterances. This strategy is exactly the same as done in the evaluation by Wo et. al [WCS19b]. Four versions for the evaluation subset of the different features Z_{CFP}, Z_{HCFP}, $Z_{|STFT|}$ and $Z_{|CQT|}$ are calculated.

MusicNet

The MusicNet dataset is split into train and test subset as suggested by [WCS19b]. The train subset contains 320 music pieces and the test subset contains 10 music pieces.

As the $\texttt{data}_{\text{CFP,Tsh}}$ version of the MAPS dataset the train subset of MusicNet are evenly cut into $\{2, 2.5, 3\}$ seconds utterances. As a textual representation REP_{Tsh} is used and Z_{CFP} is the calculated feature on this set. This results in total in 48.175 utterances.

This dataset is also evaluated with the above mentioned test subset where fixed size utterances of 2.56 seconds are extracted.

4.2 Results

In the following the results of different models are presented. In next section architectures of the models are compared. Following that a technique for overfitting is reviewed and the different preprocessing layers are evaluated. Also the different labels and features are compared and finally the best performing model is presented.

If not otherwise specified for the following evaluation results all different models are trained until epoch 100. Each epoch a checkpoint which contains the current weights of the different layers of a model is stored. After the training the checkpoints of the epoch with the lowest validation perplexity until epoch 100 is selected. The perplexity (ppl) is a value which is related to the loss which is calculated at the end of each epoch for backpropagation. The selected checkpoint is then used to load the model of that epoch and start a decoding on the test subset which has been described in section 4.1.1. The outputted sequence is then evaluated with the MIR multipitch evaluation and the resulting averaged precision $\overline{\mathcal{P}}$, recall $\overline{\mathcal{R}}$ and $\overline{F_1}$ score can be obtained in the following tables.

4.2.1 Architecture

In the following the performance of the LSTM based S2S model (in the table called S2S) and the transformer model which have both been presented in section 3.2 are compared. Also the number of encoder resp. decoder layers have been varied as depicted in table 4.3. The presented results in the table were calculated on the MAPS dataset with configuration $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$. It can be seen that the LSTM based S2S model is performing clearly worse than the transformer. For the transformer an increase of the layers from 8 to 16 results in a better result. For for the LSTM based S2S model however the F_1 score by changing the layers from 8 to 16 gets even worse.

In fig. 4.1 the ppl during the training of the different architectures can be examined. The upper plot shows the ppl during training on the validation subset and the lower plot the ppl on the training subset which is also used to calculate the gradient to backpropagate.

Architecture	Encoder layers	Decoder layers	$\overline{\mathcal{P}}$	$\overline{\mathcal{R}}$	$\overline{F_1}$
Transformer	6	6			
Transformer	8	8	51.24	55.89	53.46
Transformer	16	16	66.22	52.32	58.46
S2S	6	2			
S2S	6	2	17.76	19.03	18.37
S2S	8	8	20.63	23.14	21.81
S2S	16	16	14.83	22.96	18.02

Table 4.3: MIR multipitch evaluation for different architectures

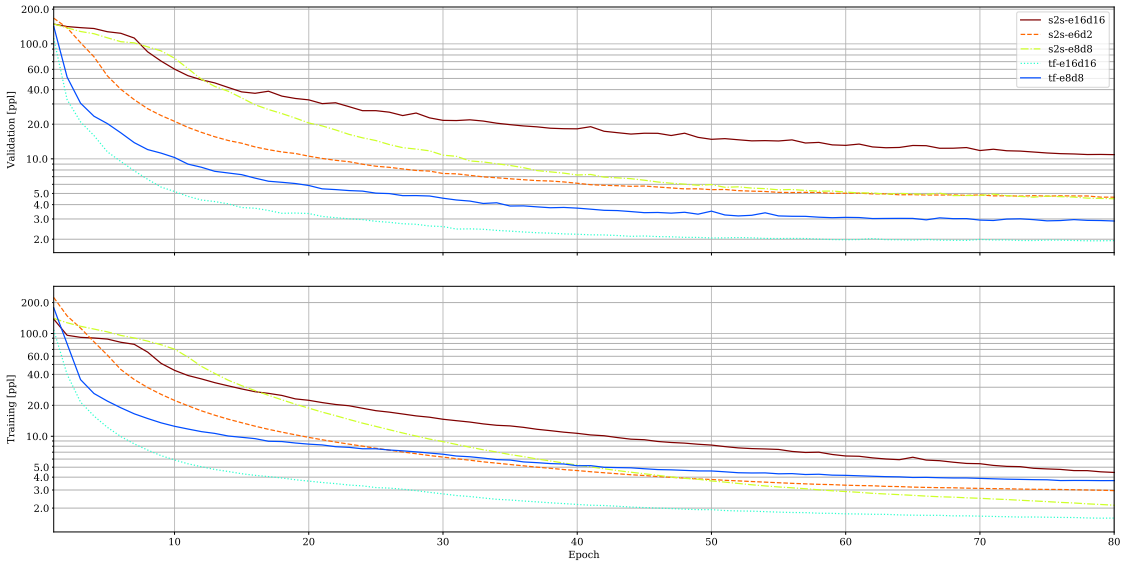


Figure 4.1: Training with different architectures

As it can be seen a lower ppl in the validation plot in that case implies that the model is more probable to perform better. It can also be seen that the 16 layer big LSTM based S2S model is converging slowly compared to the transformer with 16 layers.

Because the transformer with 16 layers performs clearly better than the LSTM based S2S architecture the following evaluation is restricted on a 16 layer encoder and decoder transformer model.

4.2.2 MusicNet

As it can be examined in fig. 4.2 for a transformer with 16 encoder and decoder layers a training procedure on the MusicNet dataset only reaches a ppl of about 4 at the validation subset. From about epoch 70 even the training ppl is only slightly dropping. Also the LSTM based S2S architecture and variations with different number of layers had a similar convergence in different experiments.

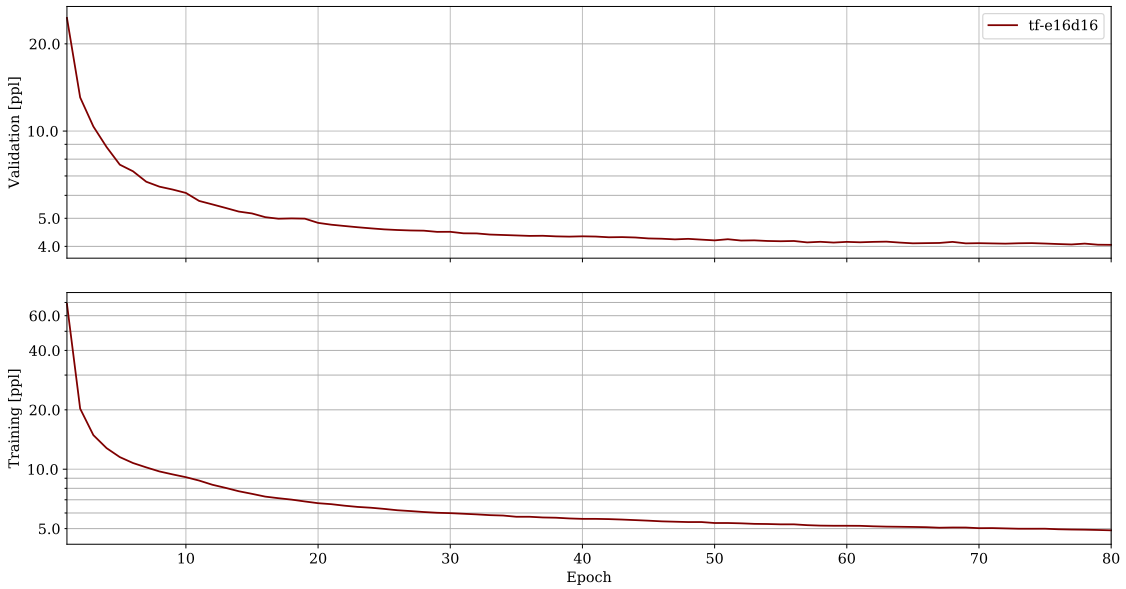


Figure 4.2: Training with MusicNet dataset

A decoding process with a ppl in this order of magnitude is ineffectual. Of course a detailed review of the dataset is complicated but accidentally found labeling problems suspect that the convergence of the training could be influenced by that. One labeling problem which has been found is that *triplet eighth* as well as *triplet sixteenth* are both labeled as *triplet*. Musically the duration of a *triplet sixteenth* is the half of a *triplet eighth*. Furthermore accidentally found last note which has been annotated in `2383.csv` is wrong annotated. This can be checked by manually listing to the recording where no note is played in the end of the corresponding file at all.

Thus in the following all analysis and comparison are only applied on the MAPS dataset.

4.2.3 Overfitting

To evaluate the effectiveness of the applied strategy against overfitting the training procedure of a transformer model is analysed. The transformer has 16 encoder and decoder layers as well as $\text{PREP}_{\text{conv}}$ as a preprocessing layer. This model is in the following called `Original`. The training has been run on the MAPS $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$ configuration.

As described in the previous section a plot of the ppl during the training of the above described model and the same model with *SpecAugment* data augmentation technique used during training is shown in fig. 4.3. The cross ($\times$) in the validation curve marks the position of the lowest ppl. It can be seen that the validation ppl of the `Original` model is not further decreasing from epoch 61 on. Whereas the validation ppl of the model with the applied data augmentation technique (`SpecAugment`) is further decreasing. That proves the technique to be effective for this training.

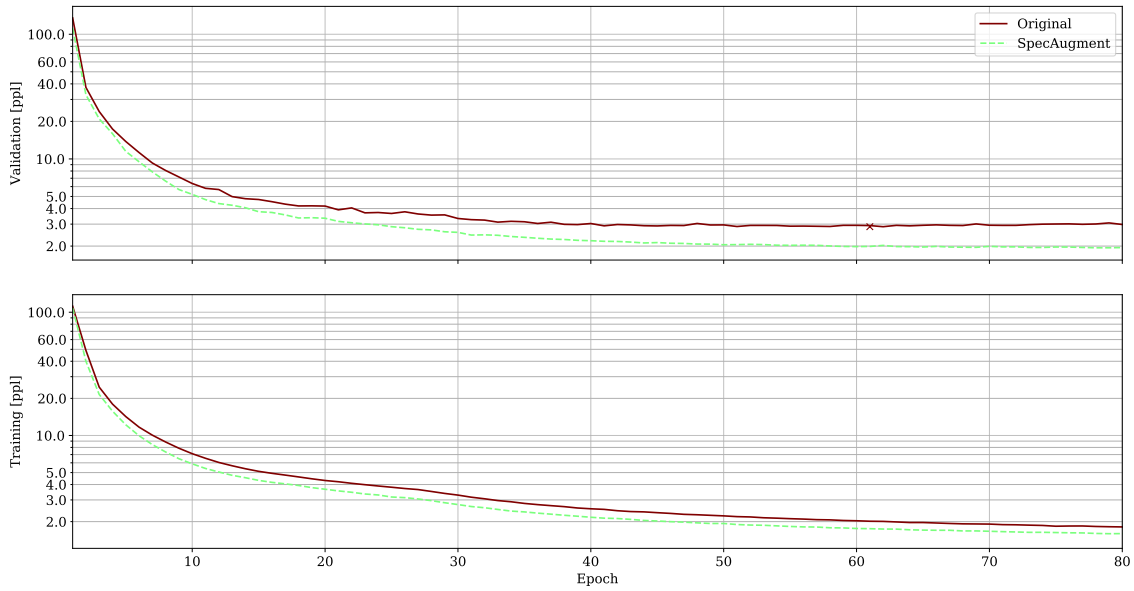


Figure 4.3: Reduce overfitting in training with *SpecAugment*

4.2.4 Preprocessing layers

To evaluate the preprocessing layers five transformer (16 encoder and decoder layers) are trained with enabled *SpecAugment* technique. Each of the transformers has a different preprocessing layer $\text{PREP}_{\text{conv}}$, PREP_{atr}, $\text{PREP}_{\text{atrn}}$ and PREP_{cf}. The training is run on the $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$ configuration for the transformer with $\text{PREP}_{\text{conv}}$, PREP_{atr} and $\text{PREP}_{\text{atrn}}$. The PREP_{cf} layer should learn the triangular filtering which has already been applied on th data in the $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$ configuration and thus the $\texttt{data}_{|\text{stft}|,\text{Tsh}}$ configuration is used instead for this training.

As it can be examined in table 4.4 all preprocessing layers but the convolutional filtering PREP_{cf} improve the overall $\overline{F_1}$ score compared to the model without a preprocessing layer (`none`). It can be seen that the batch normalisation of the atrous convolution layer of $\text{PREP}_{\text{atrn}}$ is slightly improving the $\overline{F_1}$ score compared to PREP_{atr}. The convolution layer $\text{PREP}_{\text{conv}}$ is also slights improving the $\overline{F_1}$ compared with no preprocessing layer. Maybe this can be explained by the fact that the the atrous preprocessing is directly reducing the feature by multiple convolution layers whereby for the $\text{PREP}_{\text{conv}}$ the feature size is at first increased. In $\text{PREP}_{\text{conv}}$ as recently as the embedding layer is calculated the size is reduced by a linear layer.

Maybe a joint training of the PREP_{cf} layer and the actual AMT task is causing the worse $\overline{F_1}$ score of 49.93. Another explanation could be the fact that the training with a feature size of 8000 which corresponds to the STFT bins used in $\texttt{data}_{|\text{stft}|,\text{Tsh}}$ is more difficult to learn from compared to the compact feature size of 352 from $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$.

Nevertheless in fig. 4.4 the PREP_{cf} preprocessing layer seems to learn a convolutional mapping as far as it can be interpreted in the following. In the plot monophonic chromatically played quarters are imputed to the transformer with the convolutional filtering preprocessing layer PREP_{cf} as a sound file. The upper plot shows a small part of the $X_{|\text{stft}|}$ feature

Preprocessing layer	$\overline{\mathcal{P}}$	$\overline{\mathcal{R}}$	$\overline{F_1}$
none	61.84	54.16	57.74
$\text{PREP}_{\text{conv}}$	63.43	54.95	58.88
PREP_{atr}	63.70	55.99	59.60
$\text{PREP}_{\text{atrn}}$	63.82	57.62	60.56
PREP_{cf}	53.51	46.80	49.93

Table 4.4: MIR multipitch evaluation for different preprocessing layers

calculated on the file of the chromatic notes which in total would contain as mentioned 8000 STFT bins. The result after applying the convolutional filtering is fully plotted with its 352 bins in the lower plot[1]. On a careful examination activations of different convolutional bins can be seen. Furthermore the activation changes correspond to the frequency changes which can be observed in the upper plot.

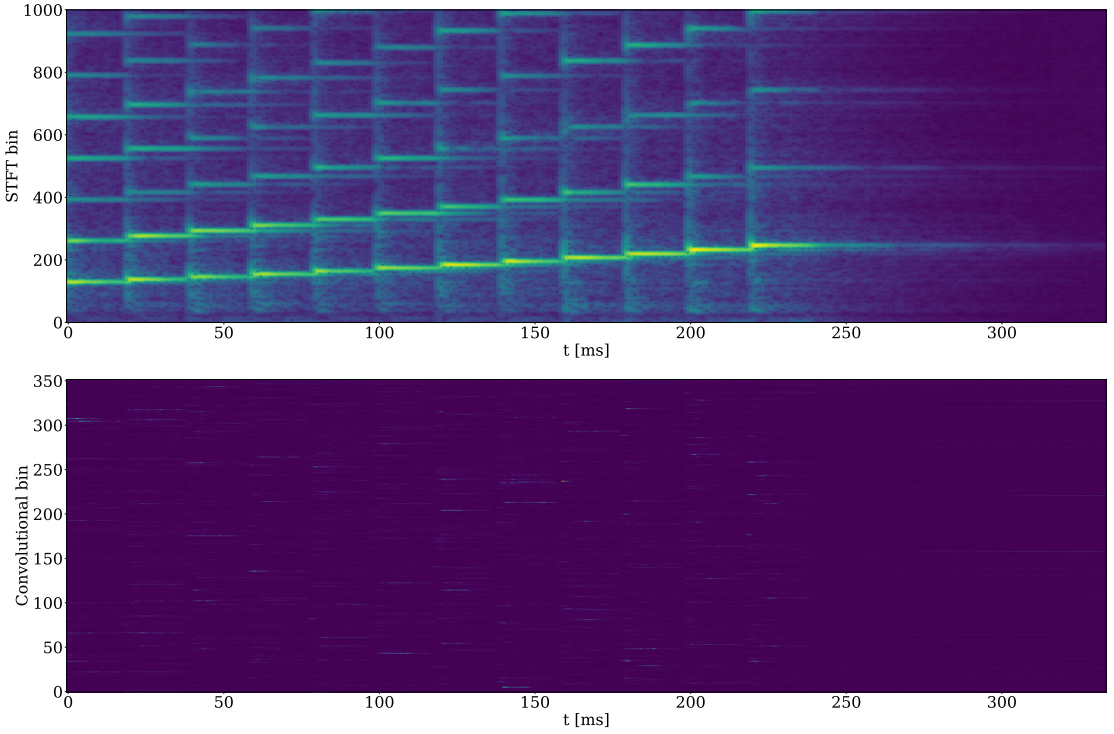


Figure 4.4: Excerpt of $X_{|\text{stft}|}$ and PREP_{cf} application on chromatic notes

It might be useful to pretrain this preprocessing layer PREP_{cf} on another dataset and finally fix the weights of the layer for a training on the actual AMT dataset MAPS.

[1]Only a softmax function per column is applied to better visualise the activations.

4.2.5 Labels

In this section both steps of the label sequencing namely the segmentation and the textual representation are evaluated. The even cutting into $\{2, 2.5, 3\}$ seconds utterances and the cutting by minimising the number of ongoing notes is compared. For the $\{2, 2.5, 3\}$ seconds cuts the textual representations REP_{Tsh}, REP_{Dis}, $\text{REP}_{\text{Dis-bpe}}$ are contrasted. In the following a transformer with 16 encoder and decoder layers as well as with the $\text{PREP}_{\text{conv}}$ preprocessing layer is used.

In table 4.5 is shown that a fixed cutting into utterances $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$ is performing clearly better than a cutting sensitive to note onsets $\texttt{data}_{\text{CFP,Tsh,min}}$. This could be explained by the fact that also the evaluation data is not cut in utterances with minimised ongoing notes. Also the REP_{Dis} representation even with the different BPEs $\text{REP}_{\text{Dis-bpe}}$ performs worse than the REP_{Tsh} representation. The number after *bpe* in the table is stating the number of iterations of token substitutions which are performed. Maybe the worse performance can be explained by the smaller sequence length of the REP_{Tsh} representation.

Dataset configuration	$\overline{\mathcal{P}}$	$\overline{\mathcal{R}}$	$\overline{F_1}$
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$			
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh,min}}$	46.34	44.98	45.65
$\texttt{data}_{\|\text{STFT}\|,\text{Dis-bpe-12}}$			
$\texttt{data}_{\|\text{STFT}\|,\text{Dis-bpe-45}}$			
$\texttt{data}_{\|\text{STFT}\|,\text{Dis-bpe-90}}$			

Table 4.5: MIR multipitch evaluation for different labels

4.2.6 Features

The three different feature which have been introduced in section 3.4 are compared on another training procedure with the transformer. This model contains the $\text{PREP}_{\text{conv}}$ preprocessing layer as in the sections before and the labels are generated corresponding to the REP_{Tsh} convention.

Table 4.6 shows that the $Z_{|\text{STFT}|}$ feature which is only based on the spectrum information is performing best. The concatenated cepstrum Z_{CFP} however reaches a better $\overline{F_1}$ score than the feature based on the CQT $Z_{|CQT|}$.

Dataset configuration	$\overline{\mathcal{P}}$	$\overline{\mathcal{R}}$	$\overline{F_1}$
$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$			
$\texttt{data}_{\text{CFP,Tsh}}$	40.90	47.03	43.75
$\texttt{data}_{\|CQT\|,\text{Tsh}}$	29.11	39.16	33.39

Table 4.6: MIR multipitch evaluation for different features

Maybe another preprocessing step is need to reduce the size of the concatenated feature Z_{CFP} which is the double of the size of $Z_{|\text{STFT}|}$.

4.2.7 Overall performance

The best performing own model considering the above mentioned evaluation for the MAPS dataset is the transformer with a $\text{PREP}_{\text{atrn}}$ preprocessing layer which is trained on the dataset with $\texttt{data}_{|\text{STFT}|,\text{Tsh}}$ configuration. Also the data augmentation strategy *SpecAugment* has been used during training. Used dropout and details about the fine tuned model are in detail printed in table 4.7.

Parameter	Value	Parameter	Value
b_input	8000	b_update	8000
d_model	512	spec_drop	True
dropout	0.2	emb_drop	0.1
freq_kn	3	freq_std	2
grad_norm	True	lr	2.5
mean_sub	True	n_dec	16
n_enc	16	n_head	1
n_warmup	8000	shared_emb	True
shuffle	True	smooth	0.1

Table 4.7: Parameters of the best performing transformer model on MAPS

In table 4.8 the best performing own model is compared to the state of the art model from Wu et al. [WCS19b]. As it can be seen the sequence to sequence learning approach in this thesis seems not as effective as the convolutional network which bases on semantic segmentation.

Model	Configuration	$\overline{\mathcal{P}}$	$\overline{\mathcal{R}}$	$\overline{F_1}$
DeepLabV3+ based [WCS19b]	-	87.48	86.29	86.73
Transformer (e16d16) with PREP_{atr}	$\texttt{data}_{\|\text{STFT}\|,\text{Tsh}}$	63.82	57.62	60.56

Table 4.8: Overall MIR multipitch evaluation

5. Conclusion

Currently this is the first S2S network used for automatic music transcription. In the thesis audio signals are preprocessed to a feature and sequentially feeded into a S2S network as a source sequence. The network learns a mapping between the source sequence and the target sequence which is a sequential musical representation.

In the part of the preprocessing different features integrating frequency, cepstrum and harmonic information have been introduced and evaluated on the model. The evaluation showed that features based on the STFT ($Z_{|\mathrm{STFT}|}$) rather than the CQT (Z_{CQT}) perform better.

Furthermore two different techniques in cutting audio files in smaller utterances have been applied. The more complex cutting at a point in time where the number of ongoing notes is minimal surprisingly showed a worse F_1 score of n points than a alternating length cutting strategy. This could be explained by the fact that also the evaluation data is cut in fixed length instead of a checking for the number of ongoing notes.

Also in this thesis different approaches for creating a sequential textual representation of the musical notation have been observed. The textual representation resulting in the shortest target sequences measured by the number of tokens outperforms the other. The short token length is achieved by intelligently ($\mathrm{REP}_{\mathrm{Tsh}}$) using tokens to describe that a musical setting is staying unchanged for a specific time.

In addition to that the evaluation showed the effectiveness of a preprocessing layer which is added in front of the embedding layer. This increases the computational ability of the model in the feature extraction and finally results in a better F_1 score.

Another essential component during the training is data augmentation. Which also for the automatic music transcription tasks evidentially reduces the overfitting to the training data. Thus also the ability of the model to generalise is increased.

Prospect

The number of utterances of the MAPS dataset can be approximately be specified as 16.500. If this is compared to the number of 4.5 million used sentence pairs in the WMT

2014 English-German machine translation task dataset [VSP+17] it can bee seen that the for this training less than 1% of data is used. As in other data driven approaches it can be assumed that a bigger number of training data could result in better performing systems. Even the type of collected music should be extended apart from classical music to also covering pop music. But the labeling could be hard for this kind of music as current music pieces are not freely available concerning the music recordings as well as sheet music.

The data augmentation strategy which has been applied to the training has not specifically been developed for the field of MIR. The results of the trained model could be improved by applying a data augmentation strategy which has been developed for musical data. Random label-preserving and pitch-shift transformations additionally changes the pitch of notes as described in [THFK18]. This enabled their model to be more robust to tuning variations in the audio files.

Another idea to improve the performance could be to integrate the time information of the labels in the training procedure more directly. For example could the attention be lead to the part where the frequencies of the token which is outputted are located. This could be done by introducing a second loss which expects the attention values where not frequency values of interest to be zero. Alternatively it could be thought of completely excluding the exact time information from the emitted tokens and deriving the exact time information from the (source) attention values.

The performance of automatic music transcription systems is increasing with current networks and improved feature extraction. This attempt showed that different subsystems and steps can be optimised which results in a better overall performance. An easy to use application based on that system can help musicians to convert played music to a written form. Furthermore the converted sound files can autonomously be analysed and improve humans musical understanding of musical recordings without the need of sheet music.

Bibliography

[AP06] S. A. Abdallah and M. D. Plumbley, "Unsupervised analysis of polyphonic music by sparse coding," *IEEE Transactions on neural Networks*, vol. 17, no. 1, pp. 179–196, 2006.

[CPSA17] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," *arXiv preprint arXiv:1706.05587*, 2017.

[DCLT18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[EBDB10] V. Emiya, N. Bertin, B. David, and R. Badeau, "Maps-a piano database for multipitch estimation and automatic transcription of music," 2010.

[HES+17] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. Engel, S. Oore, and D. Eck, "Onsets and frames: Dual-objective piano transcription," *arXiv preprint arXiv:1710.11153*, 2017.

[HKG+15] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in neural information processing systems*, 2015, pp. 1693–1701.

[HVU+18] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," 2018.

[KB14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[KR18] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[LGG+17] T.-Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[Mar04] M. Marolt, "A connectionist approach to automatic transcription of polyphonic piano music," *IEEE Transactions on Multimedia*, vol. 6, no. 3, pp. 439–449, 2004.

[Moo77] J. A. Moorer, "On the transcription of musical sound by computer," *Computer Music Journal*, pp. 32–38, 1977.

[NSNW19] T.-S. Nguyen, S. Stueker, J. Niehues, and A. Waibel, "Improving sequence-to-sequence speech recognition training with on-the-fly data augmentation," *arXiv preprint arXiv:1910.13296*, 2019.

[OSD+18] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: learning expressive musical performance," *Neural Computing and Applications*, pp. 1–13, 2018.

[PCZ+19] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[PNN+19] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Muller, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *arXiv preprint arXiv:1904.13377*, 2019.

[RMH+14] C. Raffel, B. Mcfee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. W. Ellis, C. C. Raffel, B. Mcfee, and E. J. Humphrey, "mir_eval: a transparent implementation of common mir metrics," in *In Proceedings of the 15th International Society for Music Information Retrieval Conference, ISMIR*, 2014.

[SB03] 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)*. IEEE, 2003, pp. 177–180.

[SO17] I. Simon and S. Oore, "Performance rnn: Generating music with expressive timing and dynamics," https://magenta.tensorflow.org/performance-rnn, 2017.

[SVL14] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[THFK18] J. Thickstun, Z. Harchaoui, D. P. Foster, and S. M. Kakade, "Invariances and data augmentation for supervised music transcription," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 2241–2245.

[THK16] J. Thickstun, Z. Harchaoui, and S. Kakade, "Learning features of music from scratch," *arXiv preprint arXiv:1611.09827*, 2016.

[VBB09] E. Vincent, N. Bertin, and R. Badeau, "Adaptive harmonic spectral decomposition for multiple pitch estimation," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 18, no. 3, pp. 528–537, 2009.

[VSP+17] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[WCS18] Y.-T. Wu, B. Chen, and L. Su, "Automatic music transcription leveraging generalized cepstral features and deep learning," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 401–405.

[WCS19a] M. Won, S. Chun, and X. Serra, "Toward interpretable music tagging with self-attention," *arXiv preprint arXiv:1906.04972*, 2019.

[WCS19b] Y.-T. Wu, B. Chen, and L. Su, "Polyphonic music transcription with semantic segmentation," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 166–170.

[WHH+89] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE transactions on acoustics, speech, and signal processing*, vol. 37, no. 3, pp. 328–339, 1989.

[XBK+15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *International conference on machine learning*, 2015, pp. 2048–2057.

[YK15] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *arXiv preprint arXiv:1511.07122*, 2015.

[ZLL+19] J.-X. Zhang, Z.-H. Ling, L.-J. Liu, Y. Jiang, and L.-R. Dai, "Sequence-to-sequence acoustic modeling for voice conversion," *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, vol. 27, no. 3, pp. 631–644, 2019.