University of Bergen
Department of Informatics

Automatic Drum Transcription using Deep Learning

Author: Runar Fosse
Supervisor: Pekka Parviainen

Add correct NT faculty, instead of MAT-NAT

UNIVERSITETET I BERGEN
Det matematisk-naturvitenskapelige fakultet

April, 2025

Abstract

Lorem ipsum dolor sit amet, his veri singulis necessitatibus ad. Nec insolens periculis ex. Te pro purto eros error, nec alia graeci placerat cu. Hinc volutpat similique no qui, ad labitur mentitum democritum sea. Sale inimicus te eum.

Write proper abstract

No eros nemore impedit his, per at salutandi eloquentiam, ea semper euismod meliore sea. Mutat scaevola cotidieque cu mel. Eum an convenire tractatos, ei duo nulla molestie, quis hendrerit et vix. In aliquam intellegam philosophia sea. At quo bonorum adipisci. Eros labitur deleniti ius in, sonet congue ius at, pro suas meis habeo no.

Acknowledgements

Est suavitate gubergren referrentur an, ex mea dolor eloquentiam, novum ludus suscipit in nec. Ea mea essent prompta constituam, has ut novum prodesset vulputate. Ad noster electram pri, nec sint accusamus dissentias at. Est ad laoreet fierent invidunt, ut per assueverit conclusionemque. An electram efficiendi mea.

Write proper acknowledgements

Runar Fosse
Tuesday 8th April, 2025

Contents

Chapter 1

Introduction

Within the field of Music Information Retrieval (MIR), the task of Automatic Music Transcription (AMT) is considered to be, both an important, and challenging research problem. It describes the process of generating a symbolic notation from audio. The majority of instruments are melodic, where key information for transcription would be to discern pitch, onset time, and duration. This stands in contrast to percussive instruments, where instead of pitch and duration one would focus on instrument classification and onset detection. This sets the stage for Automatic Drum Transcription (ADT), which is a subfield of AMT, specifically focusing on drums and percussive instruments. [23]

Previously, a popular approach to ADT was using signal processing, which later developed into using classical machine learning methods [23]. However in later years, deep learning has shown to be quite effective. Therfore, the recent focus of most authors has been to find the best performing deep learning approaches by either; constructing and analysing the best performing model architectures, or by finding datasets which allow models to generalize the best. [24]

Provide a good introduction into the master thesis, mentioning AMT, ADT and why deep learning is suited for such a task.

Also shortly mention how we represent the sound, and the transcriptions. What is/how do we do ADT

1.1 Thesis statement

This leads us to two primary questions. Which deep learning architecture is the best suited for solving a task like this? And, what makes a dataset optimal by making models generalize? These are two of the questions we will try to answer in this thesis.

For the former, we will train different model architectures on different, well-known ADT datasets. Specifically, recurrent neural networks, convolutional neural networks, convolutional-recurrent neural networks, convolutional transformers and vision transformers. By comparing their performances we could be able to gauge the one best suited for an ADT task.

For the latter, we will select the best performing model architecture from the first question, and train it over several different combination of the ADT datasets. By performing zero-shot evaluations, we could analyse and figure out what makes a good ADT dataset and how it would supplement a suitable model architecture.

In addition to these, we will also analyse two standard approaches when it comes to ADT and see how effective they really are, through ablation studies. These are, usage of log-filtered spectrograms, and frequency-based, dynamic timestep loss-weighting during training.

Present the aim of the thesis here. And the questions! How do we train a model capable of solving such a task at a high performing level. More specifically:

What architectures are suited for learning such a task? What datasets / combination of datasets makes the model generalize best? Of the many techniques made to help models learn this task, which ones actually help? (Ablation)

Remember the concrete What do we want to figure out.

Chapter 2

Background

2.1 Automatic Drum Transcription

As mentioned, ADT describes the task of transcribing symbolic notation for drums from audio. To be even more descriptive, ADT can be split into further tasks. From least to most complex we have: Drum Sound Classification (DSC), where we classify drum instruments from isolated recordings. Drum Transcription of Drum-only Recordings (DTD), where we transcribe audio containing exclusively drum instruments. Drum Transcription in the Presence of Additional Percussion (DTP), where we transcribe audio containing drum instruments, and additional percussive instruments which the transcription should exclude. Finally, we have Drum Transcription in the Presence of Melodic Instruments (DTM), which describes the task of drum transcription with audio containing both drum, and melodic instruments. [23]

In this thesis, we will focus on the most complex of these, namely DTM. Intuitively, we want to develop a deep learning model which, given input audio, has the ability to detect and classify different drum instrument onsets (events), while selectively ignoring unrelated, melodic instruments.

This task comes with difficulties not seen in the less complex tasks. Zehren et al. [24] describes one example, in where *"melodic and percussive instruments can overlap and mask eachother..., or have similar sounds, thus creating confusion between instruments"*.

Deep learning has shown to be a promising method to solve such a task, and several different approaches have been tried, many with great success. Vogl et al. [21, 20]

displayed good results with both a convolutional, and a convolutional-recurrent neural network. Zehren et al. [24, 25] focused on datasets, showing that the amount of data and quality of data are equally important to get good performance. Most recently, Chang et al. [6] explored an autoregressive, language model approach. This approach explored multi-instrument transcriptions, but their results on ADT were notable.

This reinforces the fact that there still exist many approaches to attempt, which could lead to a general improvement on ADT models.

2.2 The Drum Set

The drum set is a collection of percussive instruments like different drums, cymbals, and possibly different auxillary percussions. A drum set can vary in what it is composed of, however a standard kit usually consists of a snare drum, a bass drum, one or more tom-toms (toms), one or more cymbals (crash and ride), and a hi-hat cymbal [14].

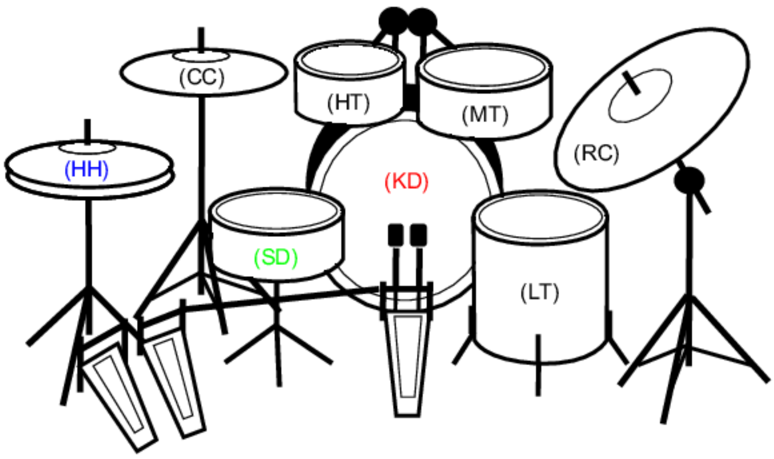

Figure 2.1: Example of the different instruments on the drumset.

As mentioned, percussion like the drum set, stands in contrast to other musical instruments in that the different ways of playing the same instrument often differ a lot in their *"audible footprint"*. The snare drum, bass drum and hi-hat all have quite different timbres, frequency span, volume, and all in all fundamentally are different instruments.

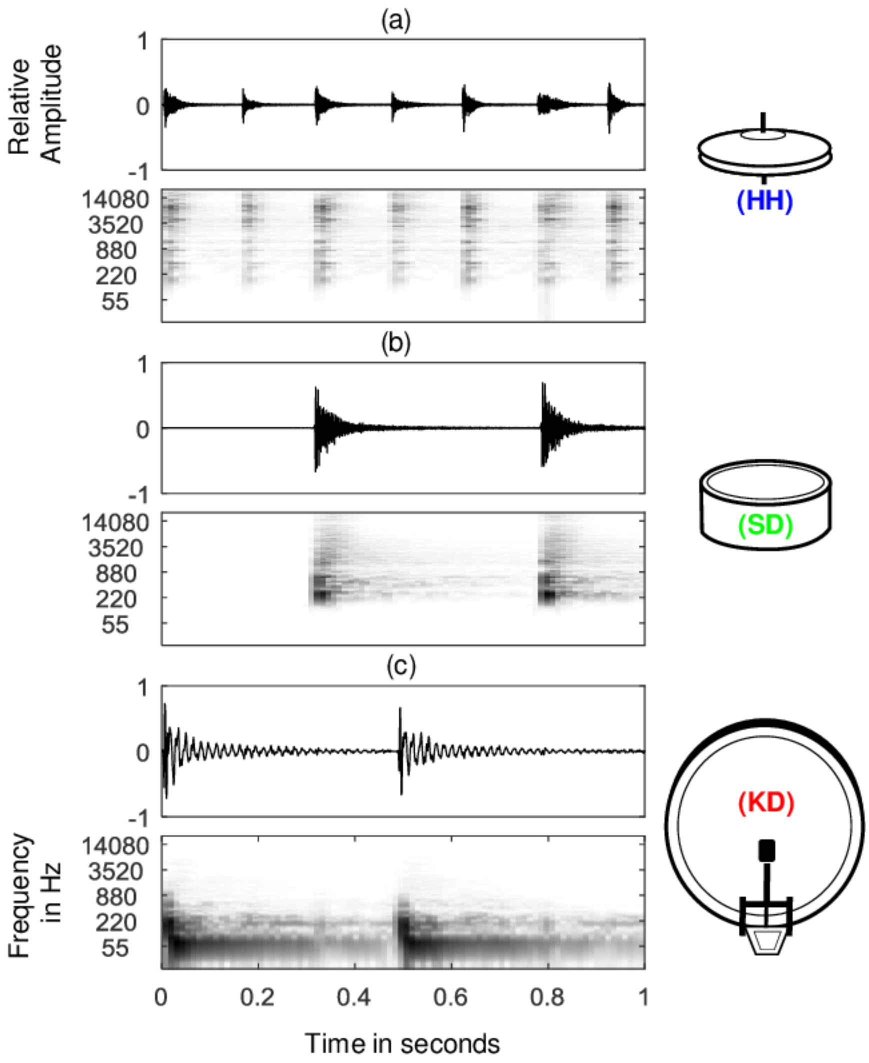

Figure 2.2: Example of the different audible footprint for drum set percussion.

Mention the different drum set instruments. Mention how they all have different musical properties, like frequency, timbre, etc (show waveforms maybe?). Also mention the most fundemental ones, and how Bass, snare and hi-hat are more important than e.g. the mid-tom or something.

2.3 Audio

Sound has be described as *"the sensation caused in the nervous system by vibration of the delicate membranes of the ear."* [1]. In short, sound is the human perception of acoustic waves in a transition medium, like air. These waves, consisting of vibrating molecules, get sensed by our auditory organs and perceived by the brain.

Thus sound can be described as the propogation and perception of waves. Mathematically, waves can be studied as signals [5]. To represent these sounds digitally, as *audio*, one can express these waves as a signal, giving rise to the *waveform*. The waveform is a representation of a signal as a graph, and charts the amplitude, or strength of the signal, over time.

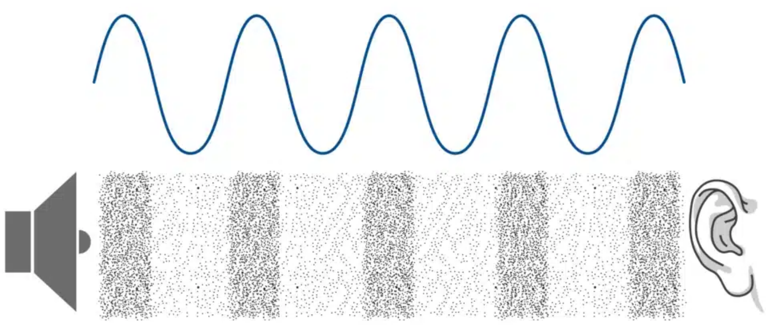

Figure 2.3: Soundwave to waveform relationship

For monophonic sound, this waveform is a one-dimensional representation. Even though this is an excellent way of storing audio digitally, it is very compact. There have been deep learning models working directly with these waveforms, e.g. Oord et al.'s WaveNet [18], however the task of parsing and perceiving such a signal is a complex one.

2.3.1 Fourier Transform

The Fourier Transform is a mathematical transformation which, given a frequency, computes its significance, or intensity, in a given signal. As we've established, audio is represented as a signal, and we can therefore use this transform to turn this audio signal into frequency space.

The fourier transform is a complex transformation. Given a signal f, we can compute the integral

$$\widehat{f}(\xi) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi\xi x} dx$$

for a frequency ξ, resulting in a *complex* number. This number consists of a *real* part and an *imaginary* part. The real part consists of the amplitude of a certain frequency, where as the imaginary part consists of the phase. This information is what allows us to, for a given signal, figure out which frequencies it is made out of and how much each frequency contributes.

By doing such a transform, we turn our temporal data into spectral data. This initively *untangles* our signal into its respective base frequencies. Such an transformation could lessen the complexity of the task, making *understanding* of audio easier.

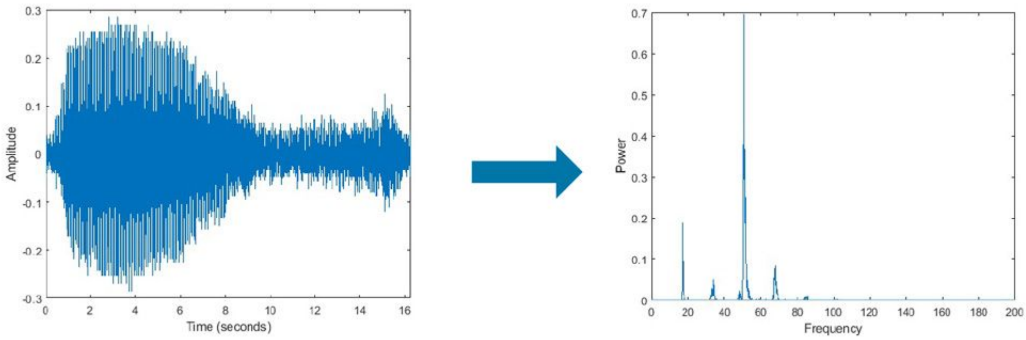

Figure 2.4: Application of a Fourier Transform

Note that the Fourier Transform is invertible, meaning that, given information about each frequency, we can perform a similar integral and reconstruct the original signal. In signal processing, this property is exploited heavily.

2.3.2 Discrete Fourier Transform

The Fourier Transform is defined as an integral over continuous time. On computers, instead of storing signals continuously we store signals using a discrete number of samples. Each signal's *sampling rate* describes how many samples a signal contains per second of audio, and is denoted in *Hz*.

To extract frequency values from these signals, we instead have to use the Discrete Fourier Tranform (DFT). Intuitively this works as the normal Fourier Transform, but ported to work on discrete-valued signals. It is given by the formula

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

where k denotes the frequency and N the number of discrete samples.

Add a example figure of FT vs DFT

2.3.3 Nyquist frequency

When we discretize a signal, e.g. when going from continuous audio waves in the air to discrete audio signals on a computer, we could lose some information. The discrete representation of the signal is an *approximation* which quality is directly dependent on the sampling rate. The higher the sampling rate, the *closer* we are to the original, continuous signal. However a higher sampling rate comes at the cost of needing to store these signals at a higher precision. A lower sampling rate would need less information stored, but this could also mean a less precise signal approxmation.

Aliasing is the phenomena where new frequencies seem to emerge in undersampled signals. For a given discrete signal, the *Nyquist frequency*, equal to half the sampling rate, is the maximum frequency a signal accurately can represent. Thus to prevent aliasing, one would need to store a signal with a sampling rate of at least double the maximum frequency.

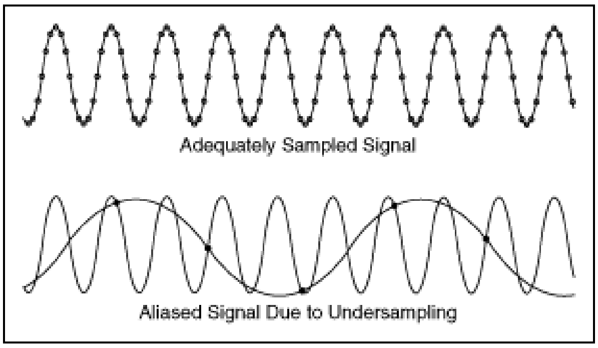

Figure 2.5: Example of aliasing in an undersampled signal.

Regarding the DFT, it here directly follows that the maximum frequency we accurately could extract information about is proportional to the sampling rate of the signal.

2.3.4 Fast Fourier Transform

Keen-eyed computer scientists may have spotted that the DFT runs in $\mathcal{O}(n^2)$ time as we, for every frequency in the range $[0, N]$ have to sum over N different values. In other words, the DFT algorithm scales quite poorly. Take into account that the standard sampling rate for audio is 44.1kHz, i.e. 44100Hz, then we can see that the DFT could be inefficient. [2]

The Fast Fourier Transform (FFT) is an algorithm which solves this problem, and instead computes the DFT of a signal within $\mathcal{O}(n \log n)$ time. Described by Gilbert Strang as *"the most important numerical algorithm of our lifetime"* [17], this practically solves our scaling problem, and allows us to efficiently extract spectral information from a signal regardless of sampling rate.

There exist many different implementations of the FFT. However the Cooley-Tukey algorithm is by far the most used FFT and optimizes calculations through a *divide and conquer* approach, utilizing previous calculations to compute others. [8]

2.3.5 Short-time Fourier Transform

The Fourier Transform comes with some drawbacks, notably how by moving from time space into frequency space, we lose temporal information. For certain tasks this might be sufficient, but the temporal dimension is vital when working with transcriptions and ADT tasks. We've seen how the Fourier Transform computes the frequencies of a signal, but what happens if we had applied the same transform to smaller, *partitions* of a signal.

This leads us to the Short-time Fourier Transform (STFT). By instead of transforming the whole signal, we transform smaller *windows*, we could gain insight into the frequency space while keeping temporal information relatively intact. This turns our data from being one-dimensional into two-dimensional, giving us insight into the intensities of different frequencies, along different timesteps.

Talk more about the partitioning. The window functions applied, and why. Spectral leakage..

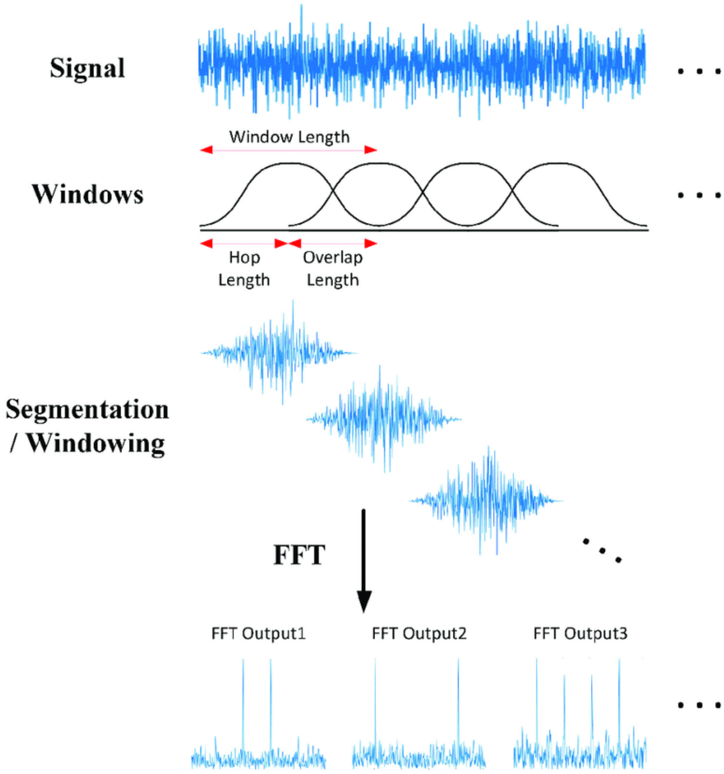

Figure 2.6: Example of the STFT

2.3.6 Spectrogram

The STFT, as the standard Fourier Transform, returns the data as complex values. To turn these into strictly real values without discarding data, we could compute the spectrogram. This is done by squaring the absolute value of each complex number.

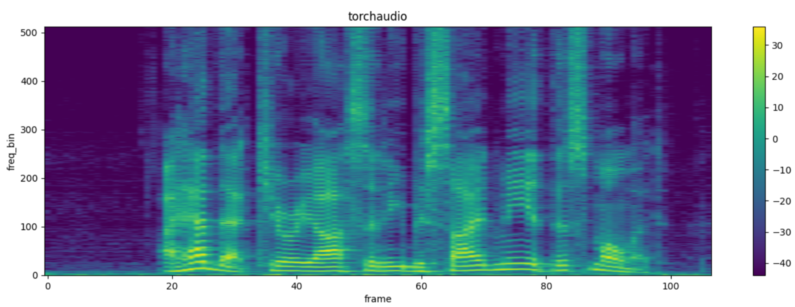

Figure 2.7: The spectrogram of an audio signal

One drawback about the spectrogram is that it contains no information about the phase of the signal it represents. That means it will not be possible to reverse the process and recreate the exact original signal. However, one could try to create an approximation like is done with the Griffin-Lim algorithm [12].

2.3.7 Filters

Signal frequencies and human perception have a special relationship. We humans percieve logarithmic differences in frequencies as a linear difference in pitch, and we tend to be better at distinguishing differences in lower frequencies than higher. E.g., the notes A_2 and B_2 have the same perceptual pitch difference as D_7 and E_7, even though their difference in frequency, $B_2 - A_2 \approx 13.471$Hz and $E_7 - D_7 \approx 287.703$Hz, are vastly different. As the frequency bins in a spectrogram are linearly spaced, this leads to the spectrogram not representing each frequency equally compared to our perception.

To solve this, we can filter the spectrogram into different bins, more suited to represent our perception of sound. This filtering is done by matrix multiplying our spectrogram with a *filterbank*; a matrix representation of different filters.

Mel Spectrograms

The mel scale, presented by Stevens, Volkmann, and Newmann in 1937, is a transformation from the frequency scale to the mel scale. These mels have the property such that a linear difference in mels are percieved as linear differences in pitch. Application of mel-filters result in the *mel spectrogram*, and are widely used when dealing with audio in machine learning, and successful applications have been seen in AMT. [22, 9, 6, 23, 11, 25]

Logarithmic Filters

The mel scale was created to mimic human perception of sound, however within ADT there is a different trend. By instead using logarithmically spaced filters, centered on the note A_4, we get a *logarithmically filtered spectrogram*. Intuitively one could assume this, instead of mimicing human perception, ports the spectrogram into a format preserving musical relationship and information. This seems to be a standard for ADT and has been used extensively by the likes of Vogl et al. [23, 21, 20, 24]

Add a example figure of Spectrogram vs Mel vs Logarithmic

2.4 Transcription

Transcription refers to a process in which we convert information from an audible format, like music, to another medium. This medium then contains a *description* of said audio. As we focus on a musical context, there are a few notable such mediums.

Explain what a transcription is, and what formats they usually are on. Explain what our model is predicting.

2.4.1 Sheet Music

Sheet music is a written transcription using musical notation that, for a given instrument, contains the *recipe* for a musician to play parts of the original recording. This is the standard when it comes to printing arrangements, and is extensively used by musicians.

Sheet music is typically descriptively exhaustive, and could contain information about musical properties like instrument onsets, tempo, velocity, etc.

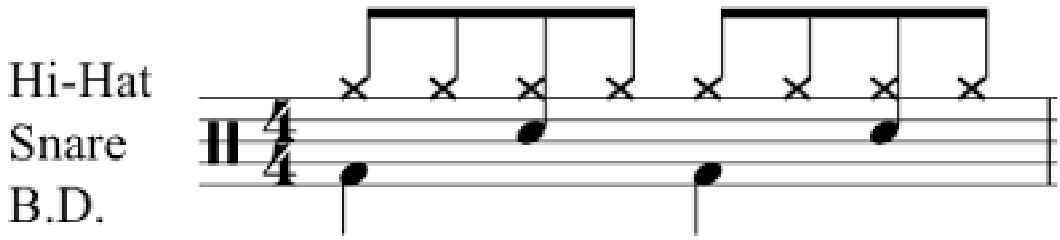

Figure 2.8: Example sheet music for a drumset

2.4.2 MIDI Annotations

Musical Instrument Digital Interface (MIDI) is the industry standard for handling music digitally. It is a binary format, containing sequences of commands that allow digital interfaces to *synthesize* music. As it is binary, it is unreadable to us humans without translating it into another format. When computers play MIDI arrangements, the MIDI sequences are parsed at a constant speed, playing different sounds through *note on*/*note off* events, delayed by time *deltas*. Similar to sheet music, MIDI is also very descriptive.

And one could say that, intuitively, MIDI is to a computer what sheet music is to a musician.

Recently, outputting transcriptions in a MIDI-like format has been attempted in DTM, and has shown to be promising. Utilizing a sequence-to-sequence Natural Language Processing (NLP) approach, Garner et al. presented MT3 [9], a model inputting spectrograms and outputting MIDI events autoregressively. This format was expanded on by Chang et al.'s YourMT3+ [6], using a Large Language Model (LLM) instead.

```
MetaEvent DeviceName SmartMusic SoftSynth 1  start : 0  delta : 0
MetaEvent SequenceName Instrument 2          start : 0  delta : 0
CC  Ch: 1 C: MAIN_VOLUME value: 101  start : 0  delta : 0
CC  Ch: 1 C: PANPOT value: 64        start : 0  delta : 0
ON:  Ch: 1 key: 67 vel: 96    start : 3072    delta : 3072
OFF: Ch: 1 key: 67 vel: 0     start : 4096    delta : 1024
ON:  Ch: 1 key: 67 vel: 96    start : 4096    delta : 0
OFF: Ch: 1 key: 67 vel: 0     start : 5120    delta : 1024
ON:  Ch: 1 key: 66 vel: 96    start : 5120    delta : 0
OFF: Ch: 1 key: 66 vel: 0     start : 6144    delta : 1024
ON:  Ch: 1 key: 62 vel: 96    start : 6144    delta : 0
OFF: Ch: 1 key: 62 vel: 0     start : 7168    delta : 1024
ON:  Ch: 1 key: 64 vel: 96    start : 7168    delta : 0
OFF: Ch: 1 key: 64 vel: 0     start : 7680    delta : 512
ON:  Ch: 1 key: 62 vel: 96    start : 7680    delta : 0
OFF: Ch: 1 key: 62 vel: 0     start : 8192    delta : 512
ON:  Ch: 1 key: 60 vel: 96    start : 8192    delta : 0
OFF: Ch: 1 key: 60 vel: 0     start : 9216    delta : 1024
ON:  Ch: 1 key: 62 vel: 96    start : 9216    delta : 0
```

Figure 2.9: Example MIDI arrangement in a readable format

Accuracy ...

2.4.3 Activation Functions

In machine learning, the task of detecting instrument onsets could be described as a multi-label sequence labeling task. This involves, for each timeframe in a sequence, predicting a probability, or rather confidence value, that a certain instrument onset happens. In the domain of MIR and AMT, it has become common place to describe these confidence distributions as *activation functions*; not to be confused with the general deep learning term, activation functions like ReLU or sigmoid. [15, 21]

This way of frame-level prediction is extensively used within onset detection in ADT and is the approach we will be taking in this thesis.

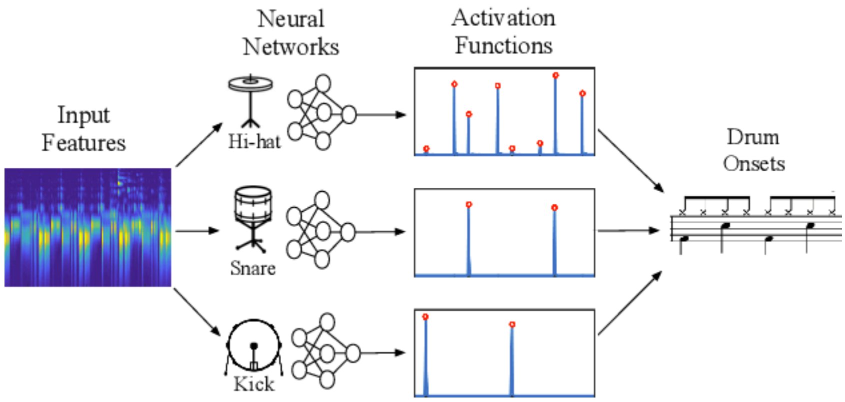

Figure 2.10: Example of ADT activation function output

Peak-picking

When predicting activation functions, we need a separate post-processing step to turn these confidence distributions into onset events. By utilizing a standard *peak-picking* algorithm, we can isolate and enhance peaks in these activation functions, and go from a continuous distribution to a collection of discrete events.

The peak-picking algorithm, introduced in its current form by Böck et al. [4], defines that a prediction $\hat{y}_n$ at timeframe n is a *peak* if it fulfills the three conditions:

$$\begin{aligned} \hat{y}_n &= \max(\hat{y}_{n-m}, ..., \hat{y}_n, ...\hat{y}_{n+m}), \\ \hat{y}_n &\geq \text{mean}(\hat{y}_{n-a}, ..., \hat{y}_n, ...\hat{y}_{n+a}) + \delta, \\ n &\geq n_{\text{last onset}} + w. \end{aligned}$$

For appropriately trained deep learning models, Vogl et al. [21] showed that the peak-picking parameters which gave the best results were $m = a = w = 2$ and $\delta = 0.1$.

2.5 Performance Measure

2.5.1 Correct Predictions

Our machine learning models predict instrument onset events on a frame-level basis. In other words, are predictions are very granular, and we need some way to decide when a prediction is correct versus incorrect. In ADT, a standard has become to allow a *tolerance window* where event predictions are correct if they lie within a certain time window, often between 25ms and 50ms. A side effect of this is that, by shifting our focus to predicted events, we lose information about *not* predicting any events [19].

2.5.2 Accuracy

For classification tasks, a standard performance measure would be *accuracy*:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}.$$

Summing up correct predictions, True Positives (TP) and True Negatives (TN), and dividing by total number of predictions, sum of TP, TN, False Positives (FP) and False Negatives (FN), we find a model's probability of having a correct prediction.

This performance measure falls short in that it is very susceptible to imbalanced datasets. In ADT, most timeframes contain no onset, meaning a naïve predictor would get a high accuracy by never predicting any onsets. Another problem with accuracy is that, due to our tolerance window approach we do not have quantities for TN, such that the standard accuracy computation is incomputable.

2.5.3 F1-score

Mentioned above are some of the reasons why *F1-score* has become the typical performance measure within ADT. F1-score combines and tries to maximize two different performance measures, namely *precision*;

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}},$$

and *recall*;

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

The precision of a model can tell us how good it is at *hitting* predictions. *Perfect precision* happens when a model has no FP, i.e. never predicting an event where one doesn't happen. Recall is similar, but represents the other end of the stick. It tells us how good a model is at *not missing* predictions. *Perfect recall* happens when a model has no FN, i.e. never *not* predicting an event where one does happen.

As mentioned, F1-score combines these two measures in an aggregate performance measure by computing their harmonic mean:

$$\text{F1-score} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.$$

By maximizing F1, we simultaneously maximize both precision and recall as well, reaping all their benefits.

2.5.4 Micro vs. Macro

There are different ways of computing and combining F1-score on multi-label data. Even though they might seem similar, they fundamentally represent different information, and thus the choice in which one to select is crucial.

Macro F1-score is computed through the arithmetic mean of the classwise computed F1-scores. Finding a model which maximizes this measure would be similar to finding the model which performes best on each of the separate classes, preventing a class from taking priority due to imbalanced datasets. Relating this to ADT, it would mean focusing on transcribing each instrument equally well.

Micro F1-score is computed through finding the F1-score with global TP, FP, FN values. Maximizing this would mean prioritizing classes that occur more frequently in the datasets. Such as in ADT, this would mean focusing on transcribing instruments which appear often, like the snare or base drum, over rarer instruments like the toms (Somewhere in this master thesis we need a quick introduction to the drums, and which piece of the drumset is which).

For ADT, the trend has been to select Micro F1-score as the main performance measure, due to its ability to show a model's *general* performance on musical pieces. We want our model to maximize their ability to transcribe music, not maximize their ability to transcribe each instrument in said music. ADT, prioritizing frequent instruments is relevant. As mentioned previously, the more frequent instruments lay the ground work for the fundamentals, and could be said to be more important than scarcely occuring ones.

Chapter 3

Architectures

Mention the different architectures, figures of their components, hyperparameters we tune on them, and motivation.

3.1 Recurrent Neural Network

The Recurrent Neural Network (RNN) is a standard architecture when it comes prediction on temporal data. It has been tried and tested, showing promising results for audio tasks.

The fundamental building block for RNNs is the *recurrent unit*. It stores information from previous timesteps in a form of memory, through maintenance of a *hidden state*.

However, traditional RNNs suffer from the *vanishing gradient problem*, making *long range dependencies* harder to learn. Different architectures have been developed to try to overcome these issues, such as the Gated Recurrent Units (GRU) by Cho et al. [7], and Long Short-Term Memory (LSTM) by Hochreiter et al. [13].

It has been shown that GRUs and LSTMs are capable of learning ADT related tasks, and is therefore in interest to comparatively measure how their efficiency stands in regards to other architectures [15, 19, 20, 24].

3.1.1 Implementation

Our RNN architecture consists of several bilateral recurrent units, ending in a framewise linear layer. For the recurrent units, we train both a GRU and an LSTM model, in addition to hyperparameter search over $L \in \{2, 3, 4, 5, 6\}$ and $H \in \{72, 144, 288\}$, selecting the one with best performance.

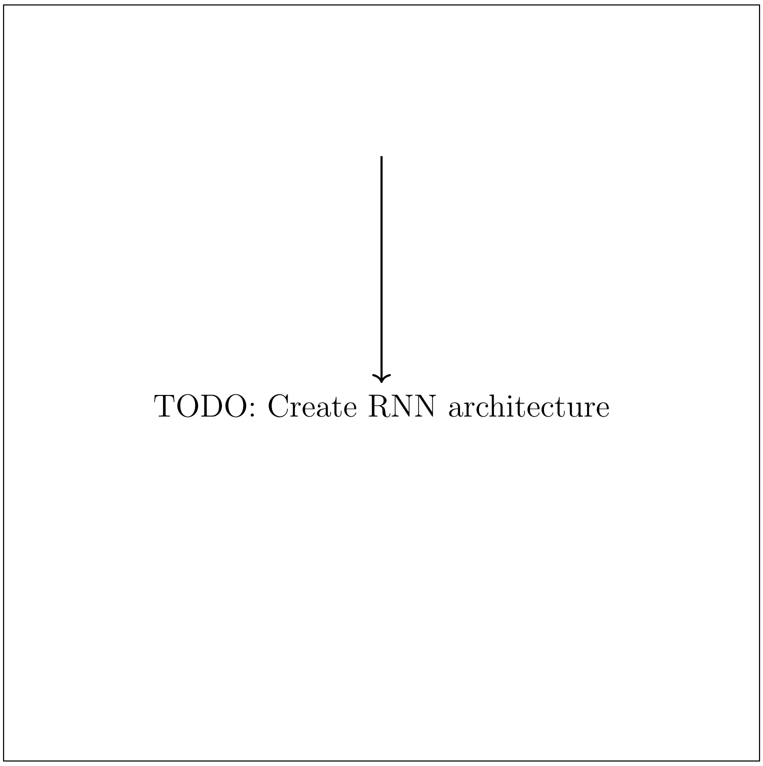

Figure 3.1: RNN architecture structure.

Show model architecture diagram like the one in ADTOF-YT paper.

3.2 Convolutional Neural Network

3.3 Convolutional Recurrent Neural Network

3.4 Convolutional Transformer

3.5 Vision Transformer

Chapter 4

Datasets

Mention the different already existing datasets used and information about them.

4.1 ENST-Drums + MDB Drums

The ENST-Drums dataset (ENST-Drums) by Gillet et al. [10] has become a staple dataset within ADT.

The same can be said for The MedleyDB Drums dataset (MDB Drums), from Southall et al. [16]. This dataset is built on top of Bittner et al.'s MedleyDB dataset [3], but re-annotated and specialized for ADT related tasks.

Due to the small size of these two datasets, they are in this thesis combined into one.

4.2 EMG-D

4.3 Slakh

4.4 ADTOF-YT

Chapter 5

Methods

5.1 Task

Precisely explain the task we are solving. Explain what the input data is, what the labels are. Give intuition into what exactly we want our model to predict.

Here we also explain the input and output, i.e. the data and the labels. What do they look like in their un-preprocessed form and predictions?

Here we can also give a figure into the pipeline itself, for better intuition.

5.2 Pipeline

Talk about the general ADT pipeline.

5.2.1 Preprocessing

Now explain what we do to the data before prediction. Explain the preprocessing steps we do afterwards (normalization, etc).

And explain how we preprocess the labels (target widening, etc.).

5.2.2 Training

Mention the loss function used, and why we use this (BCEWithLogitsLoss).

Mention the computation of infrequency weights, i.e. how they are computed, why they are computed, the intuition into how they will help us...

5.2.3 Postprocessing

Mention how model outputs a "confidence in event happening" distribution, which we want to discretize into events. I.e. explain Vogl's peak picking algorithm [21].

5.2.4 Performance Measures

Mention that we use F-measure (F1-score) is the most used and why. Compare this to accuracy, balanced accuracy. Mention precision, recall and their meaning.

Mention the difference in class-wise, micro- and macro-F1, and why we choose to focus on class-wise and micro in this thesis. Also mention how these are all computed.

5.3 Experiments

Here we mention the setups for each of the experiments.

Mention that we use RayTune to train, with PyTorch models. Mention that we only used RayTune's FIFOScheduler, and how for random search / grid search we used their built in parameter space functionality.

Mention that every single experiment was trained for at most 100 epochs, with a early stopping if validation loss didn't decrease within 10 epochs. Mention the learning rate scheduler, where we reduce the learning rate by a factor of 5 if the model hasn't improved in the last 3 epochs.

Mention that we perform early stopping on the validation loss (and why, like the smooth nature, overfitting prevention, etc.), where as we store the best performing model based on the validation F1-score (due to this representing overall prediction performance).

Mention that every experiment is model selected using hold-out validation, and best model is chosen based on micro F1-score.

5.3.1 Architecture experiment

Every sub-section below should be a separate chapter.

Shortly mention what we do, what the goal is, what we want to figure out.

Architectures

Mention the different architectures trained, and at what hyperparameters they were trained over.

Datasets

Mention the different datasets used, and tested over.

5.3.2 Dataset generalization experiment

Shortly mention why, what, like in the previous experiment.

Architectures

Mention which architectures we now use, and why we chose them. (And hyperparameters)

Datasets

Now mention which datasets / combination of datasets we use. Mention how we now use zero-shot testing (and maybe why).

5.3.3 Ablation experiments?

Technique 1

Technique 2

Technique 3

Chapter 6

Results

6.1 Architecture experiment

Display a table of results, class-wise and micro-F1: Best archicecture per dataset is bolded.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
CNN	0.5			
RNN	0.4			
Conv-RNN	0.8			
Conv-Attention	0.9			
Transformer	**0.95**			

Display the results in a barplot, to easily capture well-performing models.

Also plot enough information to be able to conclude about overall performance of models, performance on rarer instruments, etc.

6.2 Dataset generalization experiment

Display a table of results, possibly both class-wise and micro-F1 (or maybe just micro-F1): Zero-shot tests have a grayed background, best zero-shot test are bolded.

	Dataset 1	Dataset 2	Dataset 3	Dataset 4
Dataset 1	0.5	0.3		
Dataset 2	0.4	0.8		
Dataset 1+2	0.8	0.7		
Dataset 3	**0.9**	0.6		
Dataset 1+2+3	0.95	0.8		
Dataset 1+4	0.95	**0.75**		
Dataset 1+2+3+4	0.95	0.82		

Display the results in a barplot, to easily capture well-performing models.

Also plot enough information to be able to conclude about overall performance of models, performance on rarer instruments, etc.

6.3 Ablation experiments

Display results and data to be able to conclude if techniques help training / give better end results.

I.e., do we converge faster? Do we converge to a better minimum?

Could plot some loss over epochs? Need to be thorough (or average) to ensure that gains/losses are due to technique (and not hyperparameter choice, etc.).

List of Acronyms and Abbreviations

ADT Automatic Drum Transcription.

AMT Automatic Music Transcription.

DFT Discrete Fourier Tranform.

DSC Drum Sound Classification.

DTD Drum Transcription of Drum-only Recordings.

DTM Drum Transcription in the Presence of Melodic Instruments.

DTP Drum Transcription in the Presence of Additional Percussion.

ENST-Drums The ENST-Drums dataset.

FFT Fast Fourier Transform.

FN False Negatives.

FP False Positives.

GRU Gated Recurrent Units.

LLM Large Language Model.

LSTM Long Short-Term Memory.

MDB Drums The MedleyDB Drums dataset.

MIDI Musical Instrument Digital Interface.

MIR Music Information Retrieval.

NLP Natural Language Processing.

RNN Recurrent Neural Network.

STFT Short-time Fourier Transform.

TN True Negatives.

TP True Positives.

Bibliography

[1] *Fundamentals of Telephony.* United States, Department of the Army, 1953.
URL: `https://books.google.no/books?id=8nvJ6qvtdPUC`.

[2] Pras Amandine and Guastavino Catherine. Sampling rate discrimination: 44.1 khz vs. 88.2 khz. *Journal of the Audio Engineering Society*, (8101), may 2010.

[3] Rachel Bittner, Justin Salamon, Mike Tierney, Matthias Mauch, Chris Cannam, and Juan Pablo Bello. Medleydb sample, October 2014.
URL: `https://doi.org/10.5281/zenodo.1438309`.

[4] Sebastian Böck, Florian Krebs, and Markus Schedl. Evaluating the online capabilities of onset detection methods. In *International Society for Music Information Retrieval Conference*, 2012.
URL: `https://api.semanticscholar.org/CorpusID:7379180`.

[5] Pragnan Chakravorty. What is a signal? [lecture notes]. *IEEE Signal Processing Magazine*, 35(5):175–177, 2018. doi: 10.1109/MSP.2018.2832195.

[6] Sungkyun Chang, Emmanouil Benetos, Holger Kirchhoff, and Simon Dixon. Yourmt3+: Multi-instrument music transcription with enhanced transformer architectures and cross-dataset stem augmentation, 2024.
URL: `https://arxiv.org/abs/2407.04822`.

[7] Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1724–1734. ACL, 2014. doi: 10.3115/V1/D14-1179.
URL: `https://doi.org/10.3115/v1/d14-1179`.

[8] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex fourier series. *Mathematics of Computation*, 19(90):297–301, 1965. ISSN 00255718, 10886842.
URL: `http://www.jstor.org/stable/2003354`.

[9] Josh Gardner, Ian Simon, Ethan Manilow, Curtis Hawthorne, and Jesse Engel. Mt3: Multi-task multitrack music transcription, 2022.
URL: `https://arxiv.org/abs/2111.03017`.

[10] Olivier Gillet and Gaël Richard. Enst-drums: an extensive audio-visual database for drum signals processing, October 2006.
URL: `https://doi.org/10.5281/zenodo.7432188`.

[11] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer, 2021.
URL: `https://arxiv.org/abs/2104.01778`.

[12] D. Griffin and Jae Lim. Signal estimation from modified short-time fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 32(2):236–243, 1984. doi: 10.1109/TASSP.1984.1164317.

[13] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735.
URL: `https://doi.org/10.1162/neco.1997.9.8.1735`.

[14] Geoff Nicholls. *The Drum Handbook: Buying, maintaining, and getting the best from your drum kit*. San Francisco, CA: Backbeat Books, 2003.

[15] Carl Southall, Ryan Stables, and Jason Hockman. Automatic drum transcription using bi-directional recurrent neural networks. In *International Society for Music Information Retrieval Conference*, 2016.
URL: `https://api.semanticscholar.org/CorpusID:2891003`.

[16] Carl Southall, Chih-Wei Wu, Alexander Lerch, and Jason Hockman. Mdb drums: An annotated subset of medleydb for automatic drum transcription. 2017.

[17] Gilbert Strang. Wavelet transforms versus fourier transforms, 1993.
URL: `https://arxiv.org/abs/math/9304214`.

[18] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet:

A generative model for raw audio, 2016.
URL: https://arxiv.org/abs/1609.03499.

[19] Richard Vogl, Matthias Dorfer, and Peter Knees. Recurrent neural networks for drum transcription. 08 2016.

[20] Richard Vogl, Matthias Dorfer, Gerhard Widmer, and Peter Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *International Society for Music Information Retrieval Conference*, 2017.
URL: https://api.semanticscholar.org/CorpusID:21314796.

[21] Richard Vogl, Gerhard Widmer, and Peter Knees. Towards multi-instrument drum transcription, 2018.
URL: https://arxiv.org/abs/1806.06676.

[22] Friedrich Wolf-Monheim. Spectral and rhythm features for audio classification with deep convolutional neural networks, 2024.
URL: https://arxiv.org/abs/2410.06927.

[23] Chih-Wei Wu, Christian Dittmar, Carl Southall, Richard Vogl, Gerhard Widmer, Jason Hockman, Meinard Müller, and Alexander Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018. doi: 10.1109/TASLP.2018.2830113.

[24] Mickaël Zehren, Marco Alunno, and Paolo Bientinesi. High-quality and reproducible automatic drum transcription from crowdsourced data. *Signals*, 4(4):768–787, 2023. ISSN 2624-6120. doi: 10.3390/signals4040042.
URL: https://www.mdpi.com/2624-6120/4/4/42.

[25] Mickaël Zehren, Marco Alunno, and Paolo Bientinesi. Analyzing and reducing the synthetic-to-real transfer gap in music information retrieval: the task of automatic drum transcription, 2024.
URL: https://arxiv.org/abs/2407.19823.