The Development of a Drum Identifier and Transcription Tool

Louis Larsen

Advisor: Dr. Adam Finkelstein

Submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Arts in Computer Science
Department of Computer Science
Princeton University

May 2024

I hereby declare that I am the sole author of this thesis.

I authorize Princeton University to lend this thesis to other institutions or individuals for the purpose of scholarly research.

_____

Louis Larsen

I further authorize Princeton University to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

_____

Louis Larsen

Abstract

Automatic Drum Transcription (ADT) is the process of turning audio recordings of percussion instruments into another representation, typically sheet music or a MIDI file. I propose a novel approach to ADT using a CNN and a generic 2-layer Hierarchical Encoder-Decoder Transformer, which analyzes the drum audio in both the frequency and time axis. While other sub-areas of Automatic Music Transcription (AMT) use Transformers like these, ADT still relies heavily on complex, meticulously tuned CNNs. I evaluated the model with the Expanded Midi Groove Dataset (E-GMD) and IMST dataset and achieved state-of-the-art results with regard to Drum classification, Onset, Offset, and Velocity Estimations.

Acknowledgements

I would like to thank my thesis advisor, Dr. Adam Finkelstein, for his help and guidance in this paper, as well as my other graduate advisor, David Braun. Without their expertise, patience, and faith in my work, this paper would not have been possible.

I would also like to thank all my professors over the past four years, who have shaped and educated me in all areas of study, enabling me to pursue my passions for Computer Science and Music.

To Adam Ho of Tensordock, I owe my extreme gratitude for his assistance in providing me with the use of 8 NVIDIA H100 GPUs, which are some of the best available in the world right now.

Finally, I'd like to thank my friends and family for all their help and support over the last four years. Their emotional support has been invaluable and has been a constant source of inspiration, spurring me on to achieve my goals and strive for excellence.

To my Friends, Family, and Numerous Inspiring Teachers I had over the years

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Within the context of music, Automatic Music Transcription, or AMT, is the process by which a program converts a piece of music or a sound recording into a symbolic representation, typically in the form of traditional sheet music or a digital MIDI file. The process involves listening to the music and notating it to accurately represent the pitch, rhythm, dynamics, and other musical elements of the original performance. To be successful, programs must be attentive to detail, accurately capturing dynamics, articulations, and other markings. Transcription can be performed on a variety of instruments, such as pitch-based instruments like piano, violin, or guitar, or noise-based instruments like percussion. It can also be done for many purposes, such as creating sheet music for musicians to perform or identifying and analyzing chords, melody, and rhythmic patterns. [2]

For Automatic Drum Transcription, or ADT, it is important to differentiate between different noise-based instruments to understand what is playing at a certain point in an audio file. While many drum sounds are distinct, they also share similar timbral qualities. For instance, the kick drum is a drum that is played with a foot pedal, which produces a deep, low-frequency sound. The snare drum acts as the counterpart to the kick drum, which produces a sharp, cracking sound when hit with

a drumstick due to the snares stretched across the bottom of the drum. Next, a tom drum, or tom-tom, is a drum of varying sizes that combines characteristics of both a snare drum and a kick drum. Similar to a snare drum, it is struck with a drumstick, but unlike a snare drum, it does not have snares on the bottom head. This difference gives it a sound more akin to a kick drum, with a deeper, fuller, and more resonant tone. On the other end of the spectrum, cymbals produce a ringing, metallic sound when hit with a drumstick. Finally, hi-hats are controlled by a foot pedal. When the foot pedal is pressed down, it produces a clicking sound, but when the foot pedal is up, it sounds more like cymbals. [39]

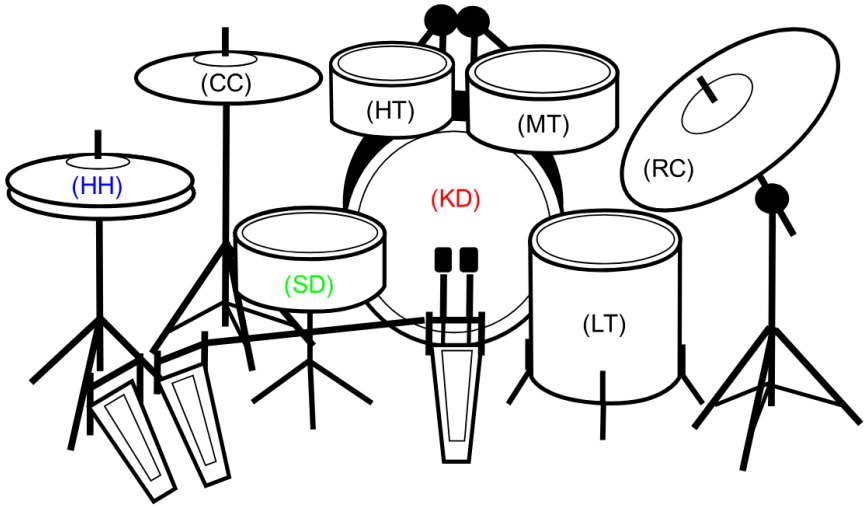

Figure 1.1: Parts of a Drum Set

Figure 1.1, adapted from Wu et al., visually demonstrates the different parts of a drum set with their abbreviations. The kick drum is labeled KD, the snare drum is labeled SD, tom drums are labeled HT (High Tom), MT (Medium Tom), and LT (Low Tom), cymbals are labeled CC (Crash Cymbal) and RC (Ride Cymbal), and the hi-hat is labeled HH. [39]

Recently, many models for ADT have used Convolutional Neural Networks, or CNNs, to help decipher the differences between these sounds. A CNN is a type of deep learning model designed for processing structured grid-like data, such as images.

It uses multiple convolutional layers to automatically learn hierarchical patterns and features from the images. Most models that use CNNs work in the following way: first, the sound of the instrument is turned into some visual format, most typically a MEL-Spectrogram, which represents the audio. A Mel-Spectrogram is a representation of the frequency content of a signal over time, where the frequency axis is converted to a logarithmic scale that approximates human hearing. Then, it studies the visual format to learn the qualities associated with each drum sound. Finally, based on those qualities, it attempts to transcribe what happens at every moment of the audio file. [6, 14, 36, 37, 38, 42] Therefore, their ability to focus on the specific defining timbral qualities of instruments combined with their study of local regions of the input make them effective at detecting sounds across different positions in music tracks.

However, while CNNs have made much progress in the field of ADT, state-of-the-art architectures that are solely comprised of CNNs have some downsides. First, these models are complex, employing many meticulously tuned hyperparameters to ensure successful transcriptions on a specific dataset. [13] Additionally, in other fields of AMT, models solely built around CNNs are no longer state-of-the-art algorithms. Instead, researchers have seen great success employing Transformer-based architectures. [10, 13, 32] A Transformer is a deep learning model architecture originally developed for sequence-to-sequence tasks and Natural Language Processing (NLP). Unlike CNNs, which provide a very localized study of audio files, Transformers rely on self-attention mechanisms, allowing them to effectively capture long-term patterns in the input data. This makes them great for tasks where understanding the context of each element in a sequence is important.

To my knowledge, an ADT model that incorporates a Transformer architecture has yet to be designed, which is strange, especially considering the role of drums in keeping rhythmic patterns in many genres of popular music. For instance, many drum patterns consist of repeating motifs that occur over several beats or measures. An

example of this would be a common rock beat, where the kick drum plays on the first and third beats of the bar, and the snare drum plays on the second and fourth beats. More complicated repeated motifs could be found in other genres like jazz or Latin music. Additionally, while these motifs may repeat, variations help keep the rhythm interesting and dynamic. These can be subtle, such as adding extra snare hits, or more elaborate, such as a drum fill that leads into a chorus or bridge, or changing the feel of the groove over time. Capturing these variations and their relationship to the underlying motifs requires understanding long-term dependencies.

Additionally, the attention mechanisms found in Transformers allow them to focus on different parts of the input sequence when making predictions. This is particularly useful in ADT, where drum sounds are overlapped or played simultaneously. By playing different drum sounds simultaneously, drummers can create rich, multifaceted rhythms that enhance the overall musical experience. Transformers can identify and process these simultaneous events, ensuring that each drum hit is transcribed accurately. This capability is essential for capturing the richness and complexity of drum patterns, enabling the model to produce more detailed and faithful transcriptions of drum performances. [39]

So, on one hand stands the established architecture of CNNs, which can learn the distinctive features of drum sounds and evaluate their specific classification at any given point in an audio file, and on the other stands the new and upcoming Transformer model, which processes long-term sequences of drum sounds to determine how each drum is expected to be heard within audio files. CNNs excel at local feature extraction, which is beneficial for identifying specific characteristics of drum hits, such as attack, decay, and timbre. However, their ability to capture long-term dependencies in sequences, especially those involving overlapping or simultaneous drum hits, is limited. This is where Transformer models offer a new approach. Transformers can process long-term sequences of drum sounds, allowing them to understand how

each drum is used within a piece of music over time. The attention mechanism in Transformers enables them to focus on different parts of the input sequence, making them well-suited for capturing the complex, polyphonic nature of drum performances.

While the two architectures seem distinct in their processing techniques, if they could learn from each other, they offer the potential to significantly improve the accuracy and detail of ADT. One could leverage the strengths of both approaches, combining both short-term features and long-term sequences, to enhance their ability to transcribe drum performances with greater fidelity.

Therefore, the main focus of this paper is to build a Transformer-based model solely for ADT that incorporates the local feature extraction capabilities of CNNs along with the attention mechanisms of Transformers. By integrating these two key components, the model aims to achieve a more comprehensive understanding of drum performances, capturing both the distinctive features of individual drum hits and the complex relationships between them over time. This integrated approach not only seeks to improve the accuracy and detail of drum transcription but also to enhance the model's ability to transcribe polyphonic drum patterns with greater fidelity. Additionally, the model's architecture will be optimized to handle the specific challenges of drum transcription, such as the overlapping and simultaneous nature of drum sounds, further enhancing its performance in this domain.

Regarding my training implementation, I plan to feed both MIDI and audio files into the model. The audio files will be chopped up and converted into Mel-Spectrograms. After going through a CNN Block to extract the most useful features from the spectrograms, the output is combined with the MIDI input labels, which are sent to a hierarchical Encoder-Decoder Transformer to study sequences and patterns in the frequency and time dimensions. The output of the Transformer is then processed through a Hidden Neural Network to detect the most probable sequence of drum events. The rest of the paper will proceed as follows: after reviewing existing

literature that delves deeper into experiments conducted in the field of AMT, I will provide a more detailed discussion of my approach, implementation, and experimental results.

Chapter 2

Background and Related Work

Research in polyphonic music transcription dates back to the 1970s when the instruments were non-percussive, and only two voices were allowed to play simultaneously. These methods used basic signal processing techniques to help transcribe music audio. [16] For instance, Piszczalski and Galler attempted to transcribe short, performed compositions on flutes and recorders using Fourier transforms and spectrogram analysis. [22] In another study, Moorer presented a program that could transcribe polyphonic music through bandpass filtering. [20]

As interest in music transcription grew in the 1990s, researchers began to explore new avenues beyond transcribing pitched music. Some delved into the complexities of percussive instruments, developing techniques for accurately capturing drum patterns and rhythms. Others focused on beat-tracking, aiming to precisely locate the beats and tempo of a piece of music or classifying musical instruments based on their sound characteristics, leading to advancements in instrument recognition technology. [16] However, this paper will focus on three background areas: Automatic Piano Transcription, Automatic Drum Transcription, and Automatic Multi-Instrument Transcription.

2.1 Automatic Piano Transcription

Developing out of the polyphonic music transcription research of the 1980s and 1990s, the field of Automatic Piano Transcription is one of the most widely focused areas of AMT today. This process involves analyzing piano audio signals to detect each note's timing, pitch, and duration.

At first, models used fundamental frequency analysis combined with either signal processing or Hidden Markov Models, or HMMs, to extract the most probable melody lines from the audio. [1, 8, 25] An HMM is a probabilistic model that describes a sequence of observable events generated by an underlying process with hidden states. The model includes parameters for state transitions, emission probabilities of observable symbols, and initial state probabilities. However, in 2006, Polliner and Ellis presented a new approach involving the use of a Support Vector Machine, or SVM. An SVM is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the hyperplane that best separates different classes in the input data space. The hyperplane is chosen to maximize the margin, which is the distance between the hyperplane and the nearest data point from each class, allowing for better generalization to new, unseen data. In their model, the SVM learns the spectral features of the piano, and using these features, the HMM transcribes the notes the piano played. As opposed to previous methods, which achieved accuracy rates of less than 50%, their model outputted the correct notes around 60 to 70% of the time. [23]

In the 2010s, neural networks started to gain prominence in the area of piano transcription. In 2012, Bock and Schedl proposed a new method of transcribing piano music using Neural Networks. Unlike previous methods, which used multiple classifiers and HMMs, their system used a single regression model, increasing the accuracy to 84%. [3] In 2016, Sigtia et al. trained several neural networks, finding that a CNN outperformed all the other neural networks. [28] In 2017, Hawthorne

et al. introduced a new model called Onsets and Frames, which used a CNN and an RNN together to jointly predict onset events and pitches, achieving significant improvements in transcription accuracy. [12]

Current state-of-the-art models use Transformers to transcribe piano audio due to their generality and ability to study sequences over time. Hawthorne et al. introduced this approach in 2021, training a generic Encoder-Decoder Transformer on both audio and midi files to achieve similar results to other top-performing piano transcription models of the time, all of which used complex, specialized deep neural networks. [13] Another model, called hFT-Transformer, was introduced by Toyoma et al. in 2023. It trains a two-level hierarchical frequency-time Transformer architecture on both audio and midi files. The output of the first layer, which studies frequency, is fed into the second layer, which studies time. [32]

2.2 Automatic Drum Transcription

Unlike Piano Transcription, Drum Transcription has received less attention from researchers. Unlike piano notes that have a clear fundamental pitch and overtone series, drum music is more noise-based, making it harder to perform frequency estimations. Therefore, drum sound transcription is much more determined by classifying one drum sound from another. However, accomplishing this task is often hard because different drum sounds can be played on top of each other. [39]

The first attempt to transcribe percussive instruments came in the mid-1980s, with researchers attempting to transcribe different types of conga strikes. However, work on transcribing drum set audio was first proposed by Goto et al., who used templates generated from spectrograms showing the frequencies of each drum type in isolation. [16] Later, in 2004, Gillet and Richard presented a new technique for transcribing drum audio. They trained both HMMs and SVMs on drum loops consisting of snare

drums, kick drums, hi-hats, and other percussive sounds. They found that SVMs were the best for transcribing these audio loops. [11] Despite this study, standard methods for drum transcription remained up in the air, with some preferring HMMs [9, 21], others preferring SVMs [30, 31], and a few even preferring k-means clustering [19, 27] or matrix factorization. [17, 26, 33, 40]

More recently, drum transcription algorithms have started to incorporate neural networks. Typically, deep learning drum transcription algorithms work in the following way. First, data is extracted from the audio using signal processing methods. Then, that data is used to train a neural network, which is then used to predict the types of drums playing at each moment of the audio file. Among the first neural networks to be implemented were Recurrent Neural Networks, or RNNs, because of their success in improving other areas of Automatic Music Transcription. RNNs are a type of neural network architecture designed to effectively model sequential data by maintaining a memory of previous inputs to inform the processing of current inputs. [29, 34, 35] However, quite soon after that, CNNs began to be used more often because they outperformed other existing methods. [6, 14, 36, 38, 42]

There are two current state-of-the-art ADT algorithms, both relying on CNN. One of the models is called DT-Ensemble, which was trained on a variety of datasets and used CNNs and CRNNs, which is a combination of a CNN and an RNN, to detect Drum Sounds. [37] The other state-of-the-art algorithm is an adaptation of Onset and Frames to Drum Audio, developed by Callender et. al. In adapting this model, they showed that the CNN model used to transcribe piano audio could also be used for drum set audio. [4]

2.3 Automatic Multi-Instrument Transcription

With the rise of deep neural networks, several researchers have attempted to transcribe multiple instruments, pitched and percussive, playing at once. To accomplish this, some models take advantage of Music Stem Separation, which is the process of isolating individual instruments from the audio recording. For example, Manilow et al. proposed a novel architecture called Cerberus, which simultaneously performs Music Source Separation on up to 5 instruments, including piano, bass, guitar, drums, and strings, and transcribes each simultaneously. [18] Another model proposed by Cheuk et al. extends this approach by performing joint stem separation and music transcription on 39 different instruments. [5]

Other models take a different approach by combining individual state-of-the-art music transcription systems into one multi-instrument transcription system. The most prominent system of this type is Omnizart, which was put together by Wu et al. It provides pre-trained models for frame-level and note-level transcription of various pitched instruments, vocals, and drum events, making it a versatile toolkit for music information retrieval tasks related to AMT. [41]

However, the current state-of-the-art model is MT3, developed by Gardener et al. Developed out of the Transformers used in current Automatic Piano Transcription models, MT3 treats Multi-Instrument Classification as sequence prediction tasks where sequences consist of tokens representing notes. To transcribe a new instrument, they simply add a new type of token. Consequently, MT3 transcribes many different types of instruments simultaneously, improving performance for low-resource instruments like guitar while maintaining strong performance for abundant instruments like piano. [10]

Chapter 3

Approach

3.1 Transformer-Based Architecture

My novel approach plans to use a Transformer-based architecture with a CNN Block to transcribe drum set Audio. Currently, recent advancements in ADT have primarily been driven by the development of architectures comprising only highly specialized CNNs. These models are meticulously tuned and refined through hyperparameter optimization to extract nuanced features from spectrograms or other visual representations of audio signals.

However, in other areas of AMT, like piano transcription and multi-instrument transcription, models solely comprised of CNNs or RNNs are no longer state-of-the-art algorithms. Instead, many models now comprise Encoder-Decoder Transformers, which treat music transcription as a sequence identification task. In succeeding with Piano Transcription, these Transformer-based models have demonstrated a superior ability to effectively capture long-range dependencies and intricate patterns in music. Additionally, Transformer architectures allow for integrating various types of information, such as looking at both midi and audio files, enabling more comprehensive music understanding and transcription.

To my knowledge, a drum set specific Transformer-based transcription model has yet to be designed. One might say that Transformer-based systems have captured drums before in Automatic Multi-Instrument Transcribers. However, while this is the case, Drum Transcription is often not the main focus of these transformer models. For instance, in MT3, drum transcription is "not a focus of this work, but [they] include them for completeness." [10] This suggests that there is still room for a Transformer model specifically designed and optimized for drum transcription tasks.

So, given the success of Transformer-based models in piano transcription and their ability to handle complex musical patterns over time, applying similar architectures to ADT shows great promise. Drum transcription, like piano transcription, requires capturing nuanced temporal and spectral features to distinguish between different drum sounds accurately. Drums produce percussive sounds with varying timbres and durations, and patterns involve syncopation, polyrhythms, fills, and a unique rhythmic feel and groove. By leveraging the strengths of Transformer models, such as their ability to capture long-range dependencies and learn from multi-modal input, the accuracy and robustness of drum transcription systems can potentially improve.

3.2 Model Configuration

My model's approach borrows heavily from other state-of-the-art Transformer models used for Piano Transcription, specifically hFT-Transformer. Figure 3.1, adapted from Toyoma et al., displays the architecture of the hFT-Transformer model. [32] My proposed method will allow spectrogram input to be converted into a MIDI output containing drum classification, onset detection, offset detection, and velocity detection. The model starts by reading in both audio input and MIDI files. The audio input is divided into specific time-frames and converted into MEL-Spectrograms. These types of spectrograms are chosen because they more closely align with human

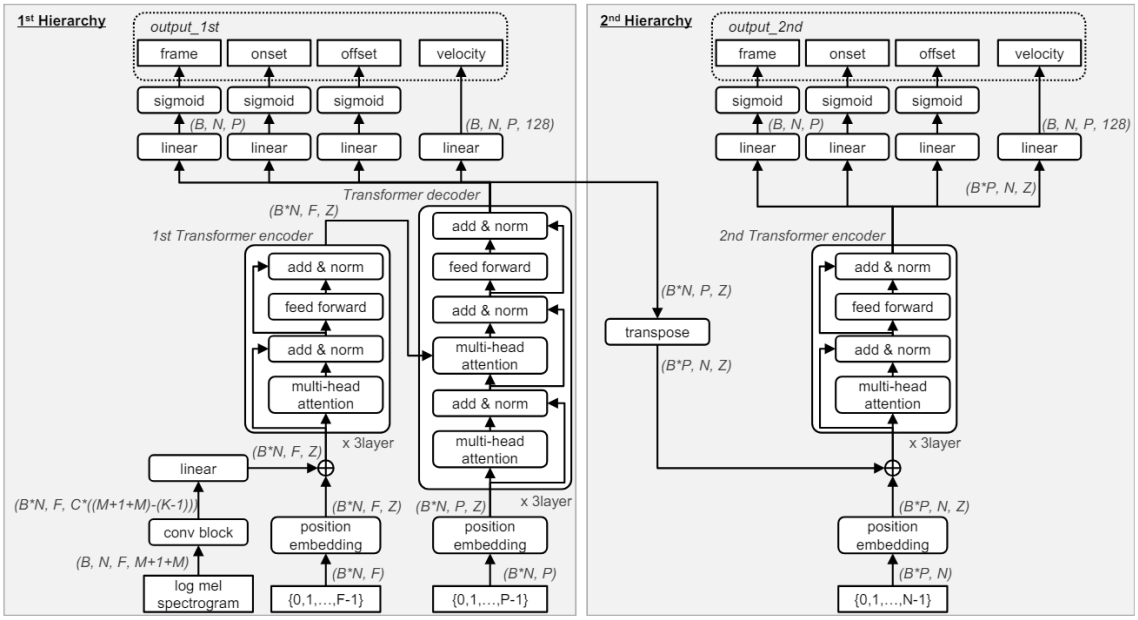

Figure 3.1: Architecture for hFT-Transformer

auditory perception. This means that MEL-Spectrograms provide a better representation of how humans perceive different frequencies. A CNN block, which performs a 1D convolution in the time dimension, is then used to extract the most important features from the generated spectrograms.

The output of the CNN block is then combined with positional embedding from the MIDI files and sent to a Transformer Encoder, which performs analysis in the frequency dimension. Here, the model studies the dependencies between the spectral features, allowing the model to learn how different musical events relate to each other in terms of their spectral characteristics and timing. Additionally, each value in the positional embedding has a trainable embedding. This process enhances the model's ability to capture the nuanced relationships between different positions in the input sequence. This allows for a better understanding of the musical context encoded in the MIDI file.

After the Transformer Encoder processes the input sequence, the output representations from the encoder are passed to a Transformer Decoder, which once again

performs analysis in the time dimension. In this Decoder, I calculate the attention between the output vectors of the first Transformer Encoder and a new set of trainable embeddings. By calculating attention between the Encoder outputs and the new embeddings, the Decoder can focus on different parts of the input sequence and incorporate relevant information for generating accurate output predictions. The decoded vectors are then sent to a hidden neural network, which processes the output of the first hierarchy.

The decoded vectors are then transposed and sent to the second hierarchical transformer. Here, the vectors are processed with another encoder in the time axis. A third trainable embedding is used. Then, like with the first hierarchy, the outputted vectors are then sent to a hidden neural network, which processes the output of the second hierarchy, predicting the instrument's classification, onset, offset, and velocity.

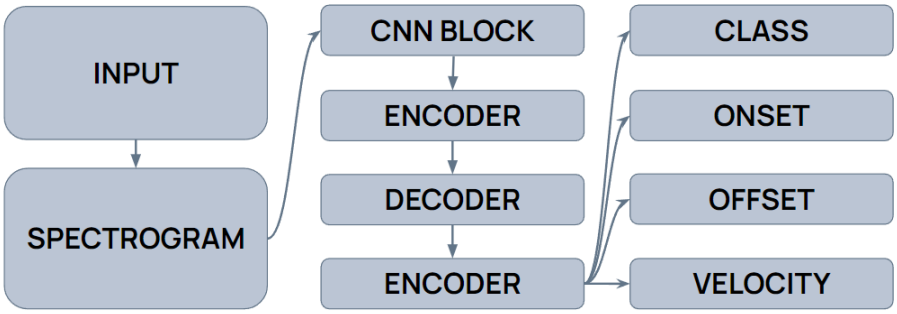

Figure 3.2: Flowchart of the Structure of My Model

By leveraging the representations learned by the CNN Block and the hierarchal transformer, the model can effectively map the complex relationships between the input musical data and the desired drum classifications and timing estimations, ultimately producing accurate and meaningful outputs for these tasks. The CNN Block extracts important features from the MEL-Spectrograms, capturing key characteristics of the audio input. These features, combined with positional embeddings from the MIDI files, provide a rich representation of the musical context. The hierarchical transformer further refines this representation, analyzing the input in both the fre-

quency and time dimensions. This allows the model to learn dependencies between different spectral features and understand how these features evolve over time. By capturing these intricate relationships, the model can accurately predict drum classifications and timing estimations. Overall, the combination of the CNN Block and the hierarchical transformer enables the model to create a detailed and nuanced representation of the input musical data. This representation is then used to make accurate predictions for drum classifications and timing estimations, producing meaningful outputs that reflect the complex nature of musical compositions.

The loss for each hierarchy is generated by summing the loss from drum classification, onset detection, offset detection, and velocity detection. For drum classification, this component of the loss function evaluates how well the model predicts the presence or absence of each drum in the input. It compares the predicted drum activations to the ground truth labels, assigning a higher loss for incorrectly predicting the presence or absence of drums. For onset detection, it measures the discrepancy between the predicted onset times and the actual onset times of drum events. Similarly, offset detection evaluates how accurately the model predicted the offset times of drum events. Finally, the loss for velocity detection assesses the deviation between the predicted velocity values and the true velocity values of drum hits. By summing the losses from these four tasks, the loss function provides a comprehensive measure of how well the model performs across all aspects of drum transcription. The model's total loss is determined by summing the loss from each hierarchy. Though both outputs from the first and second hierarchy were used to calculate the loss, only the outputs from the second hierarchy are used as the overall output.

This approach to calculating the loss for drum transcription tasks is beneficial for several reasons. Firstly, by incorporating multiple components such as drum classification, onset detection, offset detection, and velocity detection into the loss function, the model is encouraged to learn a more comprehensive representation of

drum events in the input. This helps capture the nuances of drum performances, such as the timing, duration, and intensity of drum hits, leading to more accurate transcriptions. Secondly, by summing the losses from these different tasks, the model is trained to balance its performance across all aspects of drum transcription. This prevents the model from focusing too heavily on one aspect at the expense of others, ensuring a more balanced and accurate transcription overall.

Chapter 4

Implementation

4.1 Dataset

I use the Expanded MIDI Groove Dataset (E-GMD) for the training and evaluation processes. This dataset comprises 444 hours of drum set audio from 43 different Drum Kits, ranging from acoustic to electric sounds. All sounds were recorded on either a Roland TD-11 or TD-17. I used the train/validation/test split configurations as provided. There are 35,217 sequences in the training set, lasting 341.4 hours, 5,031 sequences in the validation set, lasting 52.2 hours, and 5,289 sequences in the testing set, lasting 50.9 hours. [4] More information can be seen in Table 4.1.

Split	Unique Seq	Total Seq	Duration
Training	819	35217	341.4hrs
Validation	123	5289	50.9hrs
Testing	117	5031	52.2hrs
Total	1059	45537	444.5hrs

Table 4.1: Dataset Distribution in E-GMD Dataset

Additionally, I will be using the IMST dataset for evaluation to compare it to other models. The IMST dataset is a medium-sized dataset consisting of 104 polyphonic drum set recordings with labeled onsets in 3 different drum types: snare drums, kick drums, and hi-hats. It is often used as a standard to compare different models against

each other. [7]

4.2 Model Configuration

When the drum samples are read in, they are converted to mono. Then, the resampled waveform is transformed into a MEL-Spectrogram. To create the spectrograms, I used the transforms.MelSpectrogram class in the torchaudio library. [43] The sample rate used was 44100 hz. This was chosen to allow the model to better process events with higher frequency content, like cymbals. The window size, or the length of each segment to analyze, was 2048 samples, while the hop length, or the number of samples between successive frames in a spectrogram, was 256. This resulted in a hop size of 5.8ms. The size of the Fourier Fast Transform was 256, padding was set as constant, and Hann windows were used. Regarding the number of MEL frequency bins, 128 was used because it is a reasonable setting that preserves the timbral characteristics of the audio. An example of spectrogram input can be seen in Figure 4.1. The upper picture represents a normal input. On the bottom, the data has been square-rooted for visualization, helping to enhance the visibility of low-intensity features.

The design of the CNN block is as follows: there are 4 channels, 4 layers, and a 5×5 kernel to study the spectrogram. Both stride and padding are set to 1. Each layer consists of a 2D convolution with batch normalization, ReLU activations, and an average pooling function with a kernel size of (1, 2), stride of (1, 2), and padding of 0. The first layer has 48 filters, the second has 64, the third has 96, and the fourth has 128. The output of the CNN is embedded with the corresponding information from the MIDI files. The embedding vector size is 256.

In my work with Transformers, I utilize a feedforward neural network (FFNN) with a size of 512. This FFNN is integrated into a larger Transformer architecture, which includes four attention models, or heads, to capture different input data aspects.

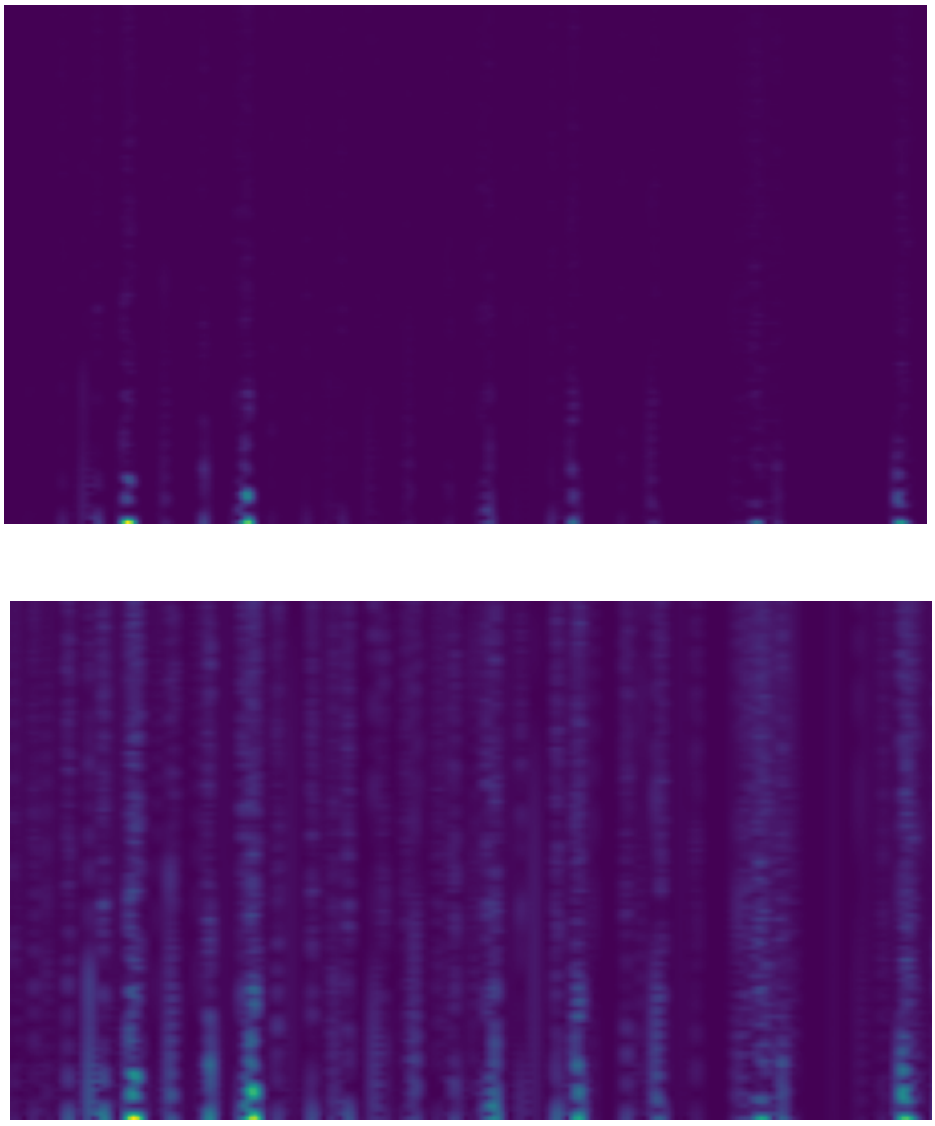

Figure 4.1: Example Mel-Spectrogram Input

Additionally, both the encoding and decoding layers of the Transformer consist of three layers each, contributing to the model's ability to learn complex patterns and dependencies within the data. This configuration allows the Transformer model to effectively process and generate sequences, making it very suitable for the area of drum transcription.

Regarding loss, Binary Cross-Entropy Loss was used for Drum Classification, Onset Detection, and Offset Detection, and 128-category Cross-Entropy Loss for Velocity Detection. The categorical loss was used for velocity rather than linear loss because researchers who made the original hFT-Transformer found that this yielded "much better performance than the MSE from a preliminary experiment." [32]

For training, I trained my model for 20 epochs using 8 NVIDIA H100 SMX5 GPUs. To allow my model to be trained on 8 GPUs, I used Pytorch's Distributed Data Parallel System. I used a batch size of 40 and a learning rate of 0.0001. I chose to use the Adam Optimizer [15] for its effectiveness in training neural networks. A dropout rate of 0.1 is applied to the model, which helps prevent overfitting by randomly dropping out connections during training. Additionally, I use a clip norm of 1.0 to prevent the gradients from becoming too large, which can lead to unstable training. For learning rate scheduling, I use the ReduceLROnPlateau callback from PyTorch, which reduces the learning rate when a metric has stopped improving, allowing for finer adjustments during training. The best model was determined by picking the one with the highest F1 scores based on a random subset of the validation dataset.

To evaluate my model, I will employ standard metrics, including accuracy, precision, recall, and F1 scores. These metrics are calculated using the mir_eval Python library [24], which is commonly used in music information retrieval tasks. Each recording is assessed individually, and at the conclusion of the testing phase, the scores for all recordings are averaged to derive the final performance metrics for the

model. This approach provides a comprehensive evaluation of the model's performance across a range of metrics, ensuring a thorough assessment of its effectiveness in the task at hand.

Chapter 5

Evaluation

5.1 Training and Validation Loss

First, Figure 5.1 shows the training and validation losses over each epoch of the model. The training loss, shown in blue, steadily decreases over epochs, indicating that the model is learning from the training data. However, the validation loss, depicted in orange, initially decreases but then starts to increase slightly after a certain number of epochs. This divergence between the training and validation loss suggests that there may be some degree of overfitting, as the model is performing better on the training data compared to the validation data. However, the relatively small gap and the overall decreasing trend in both losses indicate that the model is generalizing well to the validation data. Overall, these training and validation losses indicate that the model is learned from the training data. By the end of the training process, the training loss dropped to less than 0.15, while the validation loss stayed stagnant at around 0.25.

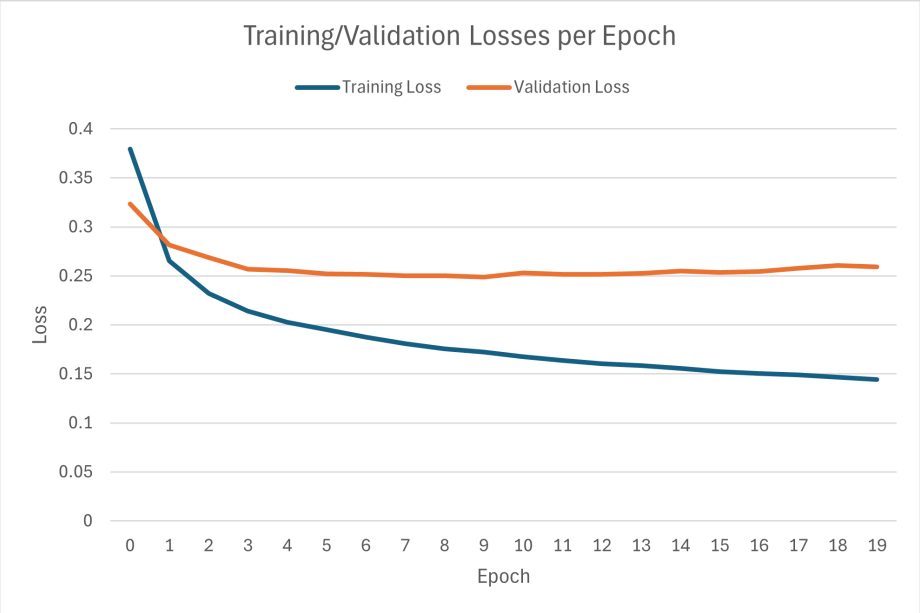

Figure 5.1: Training and Validation Loss

5.2 Best Model Selection

As stated previously, the best model was chosen based on the one with the highest F1 Score when testing with the validation dataset. All F1 scores for my model seemed to peak at around Epoch 16. In Figure 5.2, we see the F1 Score, Precision, Recall, and Accuracy for the Drum Classification output of the model. While scores start low at around 70%, one can see that the highest F1 score of 89% is achieved in Epoch 16. Figure 5.3 shows the F1 Scores, Precision, and Recall for exclusively onset detection. Once again, while scores start low around 55%, the best F1 score of 93.5% is achieved in Epoch 16. Similarly, in Figure 5.4, the F1 Score for Onset with Offset Detection seems to peak with a score of around 91.5%. Finally, in Figure 5.5, the F1 Score for Onset with Offset and Velocity Detection peaks in Epoch 16 with a score of around 88%. Overall, these results indicate that Epoch 16 represents a critical point where the model's performance reaches its peak for multiple evaluation metrics and tasks, highlighting the importance of this epoch in model training.

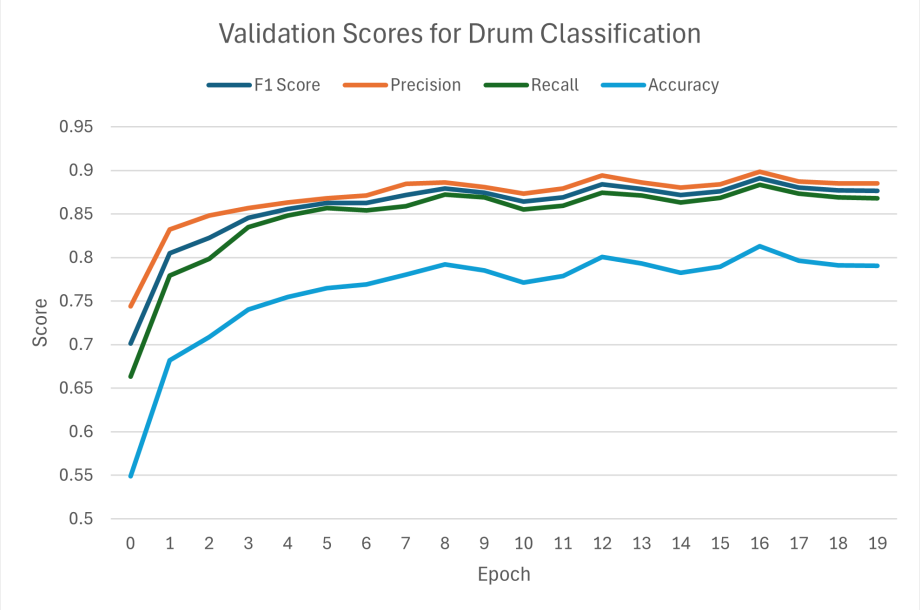

Figure 5.2: Drum Classification Validation Score

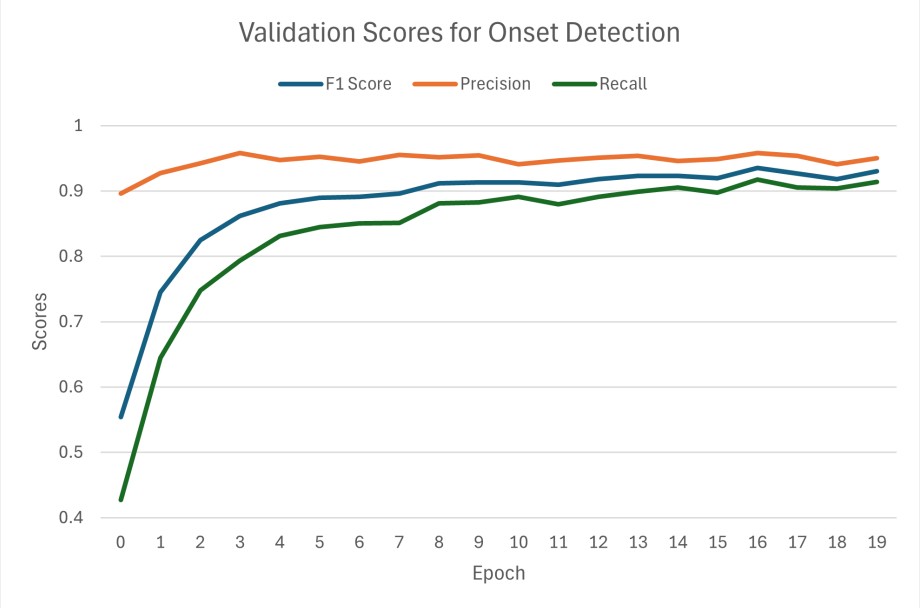

Figure 5.3: Onset Detection Validation Score

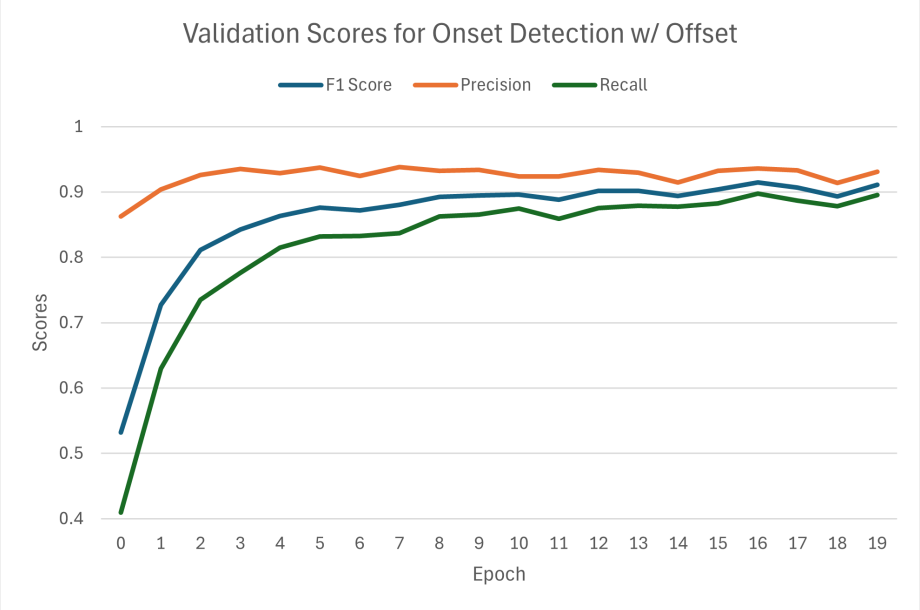

Figure 5.4: Onset and Offset Detection Validation Score

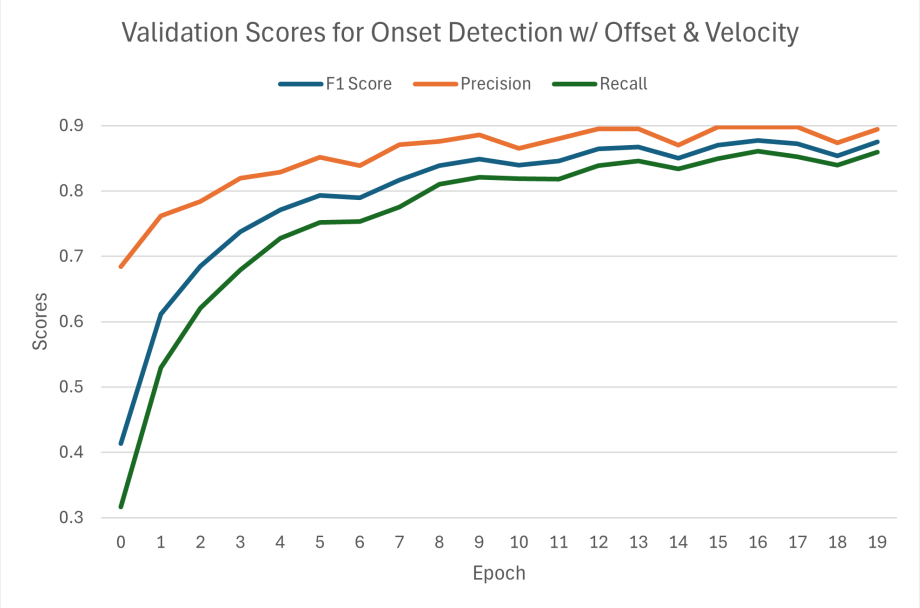

Figure 5.5: Onset, Offset, and Velocity Detection Validation Score

5.3 Testing Results

	Precision(%)	Recall(%)	F1 Score(%)	Accuracy(%)
Drum Classification	85.75	82.82	83.28	73.91
Onset Detection	87.96	84.43	86.02	—
Onset w/ Offset	85.53	82.15	83.67	—
Onset w/ Offset & Velocity	82.64	79.40	80.86	—

Table 5.1: Evaluation Results on the E-GMD Dataset

Table 5.1 shows the evaluation scores on the test set of the E-GMD dataset. Overall, the model performs well across all evaluated tasks, with precision, recall, and F1 scores ranging from approximately 80% to nearly 88%. These metrics indicate that the model achieves state-of-the-art results and can effectively classify drum sounds, detect onsets, and perform onset-related tasks with offset and velocity estimation.

5.4 Comparisons with Other Models

Model	F1 Onset Score (%)	F1 Onset w/ Velocity Score (%)
My Model	86.02	83.11
OaF-Drums	83.4	61.70
DT-Ensemble	64.98	—

Table 5.2: Onset and Onset w/ Velocity Comparison Evaluation Results on E-GMD dataset

Table 5.2 compares the onset and onset with velocity evaluation F1 scores between my model and other current state-of-the-art ADT models on the E-GMD dataset. Scores for the other models come from Callender et al. [4] My model stands out by achieving the highest F1 scores for both Onset Detection and Onset with Velocity Detection. This indicates that my model performs exceptionally well in accurately detecting the onsets of drum sounds and estimating their velocities compared to the other models. The chart accompanying the table highlights a significant improvement in velocity estimation. This improvement suggests that my model is very effective at

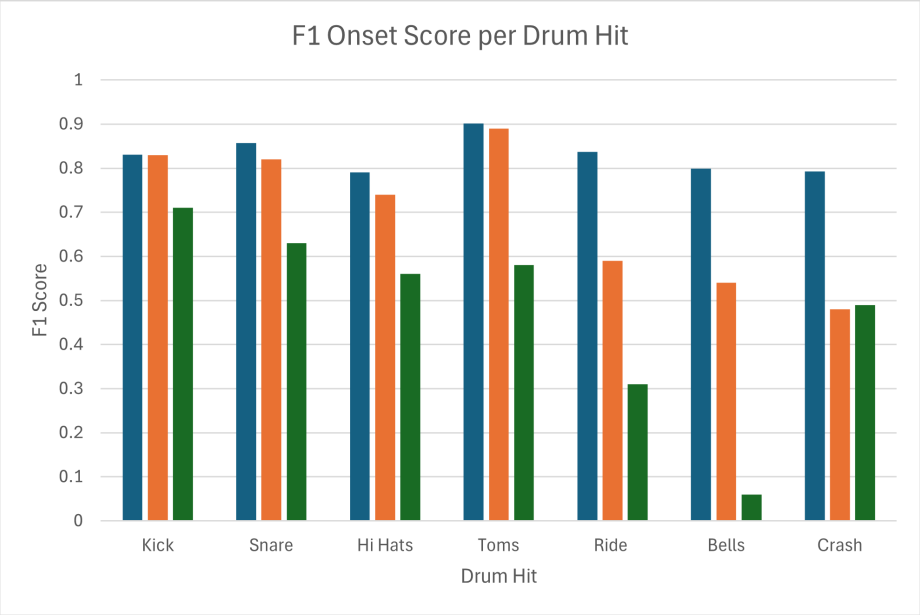

Figure 5.6: F1 Scores per Drum Classification evaluated on E-GMD

capturing the nuances of drum performances, such as the intensity or force of each drum hit, which is crucial for accurate music transcription.

Figure 5.6 shows the F1 scores for the onset detection from my model as well as other state-of-the-art models of various drum hits such as Kick, Snare, Hi Hats, Toms, Ride Cymbals, Bells, and Crash Cymbals. Scores for those models come from Callender et al. [4] The chart indicates strong performance across the board for my model, indicating big improvements with Ride Cymbals, Bells, and Crash Cymbals. This suggests that my model has robust onset detection capabilities in these areas, especially when compared to other prominent models.

Model	F1 Onset Score (%)
My Model	77.90
OaF-Drums	85.72
DT-Ensemble	91.49

Table 5.3: Onset Evaluation Results on IMST dataset

Finally, Table 5.3 compares the onset evaluation F1 scores between my model and other state-of-the-art models on the IMST dataset. Scores for those models come

from Callender et al. [4] Here, while my model achieves good performance, it does not achieve scores as good as the other prominent models. This suggests that my model may have a generalization problem, and it may excel in certain contexts or datasets while lagging behind in others.

Chapter 6

Conclusions and Future Work

6.1 Challenges

During the implementation of this project, I faced many difficulties in achieving these results. For instance, when I first started training the model, I attempted to overfit it on a small dataset. However, there was a big problem. The model could never overfit because the recall values always seemed to be low when I evaluated it on the training dataset. This indicated a high false negative rate. Drums that should have been detected were commonly not being detected. After some research, the model was only picking up on one type of drum, the kick drum, and ignoring everything else.

For months, I dedicated significant effort to enhancing my model's recall value. I explored various strategies, such as adjusting the size of the model by expanding and reducing its complexity, rebuilding the corpus data used for training, and experimenting with dropping the evaluation threshold to make the model more sensitive to certain patterns. Despite these endeavors, the improvements in recall were minimal.

At one point, I got frustrated and decided to run the overfitting tests on a piano dataset since that was what the model I adapted for ADT was initially trained on. If

the model had succeeded in this test, I would have had to rewrite my model. However, this was not the case. The original model performed far worse than my model at the overfitting test. This meant the model was too complex to pick up any information from a small dataset.

With this in mind, I went ahead and started the training process. I figured that the GPU on my personal computer, an NVIDIA RTX 3070, would be strong enough to process the data. However, after running the code for roughly half a week, one epoch had not even been trained. This unexpected delay underscored the immense computational demands of the task at hand, highlighting the need for more robust computational resources to effectively train the model in a timely manner.

Consequently, I purchased a server with an NVIDIA A100 80 GB GPU. This cut the training time one epoch from more than four days to around 30 hours. Over the next two weeks, I managed to train eight epochs. However, during training those epochs, I noticed that my learning rate scheduler had a bug and that the sample rate was improperly configured for drum set Audio. After fixing the bug and regenerating the spectrograms with the higher sample rate, I located a company that would give me access to 8 NVIDIA H100 80 GB GPUs. When these GPUs were networked together, they significantly boosted training time, reducing the training time per epoch to 2 hours. This allowed me to complete 20 rounds of training in under two days. Those models appear in this paper.

6.2 Limitations

While my model's F1 scores were the highest performance ever achieved on the E-GMD dataset, they were much lower when evaluated on other datasets. This suggests that my model may have a generalization problem. In other words, my model performs extremely well on the E-GMD dataset, but those results might not be transfer-

able to other datasets. Future research must be done to improve the generalization of my model, but a great place to start would be to analyze the onset precision and recall values on the IMST dataset and compare them to those same values in the E-GMD dataset.

The onset precision and recall values generated from the IMST dataset were 74% and 85.39%, respectively. Compared to the same values on the E-GMD dataset, recall values are the same, but precision drops significantly. This drop in precision is a significant finding, indicating that the model's ability to correctly identify onset detections relative to all instances classified as positive decreases when applied to the IMST dataset. This discrepancy in precision values between datasets highlights the dataset-specific challenges and characteristics that can impact the performance and generalization of the model. Further investigation into why this drop in precision on the IMST dataset could provide valuable insights for improving the model's performance across different datasets.

Another significant limitation of my ADT model is its high computational intensity, primarily due to the large number of parameters used in it. This high computational demand can lead to longer processing times and requires powerful hardware, such as GPUs or TPUs, to run the model efficiently.

Finally, the model's dependency on isolated drum set audio poses another limitation. It can only transcribe drum sounds accurately when the drum set audio is separated or isolated from the rest of the audio signal. This means that to transcribe songs or other musical pieces containing drums, a pre-processing step is required to isolate the drum audio, which adds complexity and limits the model's applicability to real-world scenarios where drum audio may not be readily separable.

To address these limitations, future improvements could focus on optimizing the model's architecture to reduce computational complexity, exploring techniques for drum source separation within the model itself, or developing pre-processing methods

that can reliably isolate drum audio from mixed audio signals. Methods could also be developed to transcribe drum audio when other instruments are present. These enhancements would make the ADT model more efficient and applicable to a wider range of audio recordings without the need for prior drum isolation.

6.3 Breakthroughs

Overall, my model demonstrates significant promise in the field of Automatic Drum Transcription (ADT). By employing a novel approach that incorporates Convolutional Neural Network (CNN) output of spectrograms into a hierarchical Transformer architecture, the model achieves impressive F1 scores ranging from 80% to 90% across various ADT tasks when evaluated on the E-GMD dataset. Particularly noteworthy are the substantial improvements observed in transcribing drum sounds with velocity information and accurately detecting Cymbal onsets, indicating the model's effectiveness in capturing nuanced aspects of drum performances.

In the future, this project has the opportunity to open up many different creative opportunities for musicians. This project could revolutionize how drummers interact with and create music. For instance, musicians could use the transcribed drum tracks as a foundation for composing new pieces, experimenting with different drum patterns, or analyzing and improving their own performances. Furthermore, the ability to transcribe drum performances with high accuracy and detail, including velocity information, opens up possibilities for creating more realistic and expressive drum tracks in digital music production. Overall, this project has the potential to not only streamline the music creation process but also inspire new forms of musical expression and creativity.

But for now, despite these achievements, there remains room for improvement. The model's performance on the more challenging IMST dataset, where it exhibited a

decline in precision compared to the E-GMD dataset, highlights the need for enhanced generalization and robustness. Further optimizations could be explored to reduce the model's computational intensity, making it more practical for real-time applications. Continued research and development efforts are essential to address these challenges and further enhance the model's performance in ADT tasks, solidifying its position as a valuable tool in music transcription and analysis.

Bibliography

[1] J. Bello, L. Daudet, and M. Sandler. Automatic piano transcription using frequency and time-domain information. *IEEE Transactions on Audio, Speech, and Language Processing*, 14(6):2242–2251, 2006.

[2] E. Benetos, S. Dixon, Z. Duan, and S. Ewert. Automatic music transcription: An overview. *IEEE Signal Processing Magazine*, 36(1):20–30, 2019.

[3] S. Böck and M. Schedl. Polyphonic piano note transcription with recurrent neural networks. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 121–124, 2012.

[4] L. Callender, C. Hawthorne, and J. H. Engel. Improving perceptual quality of drum transcription with the expanded groove MIDI dataset. *CoRR*, abs/2004.00188, 2020.

[5] K. W. Cheuk, K. Choi, Q. Kong, B. Li, M. Won, J.-C. Wang, Y.-N. Hung, and D. Herremans. Jointist: Simultaneous improvement of multi-instrument transcription and music source separation via joint training, 2023.

[6] K. Choi and K. Cho. Deep unsupervised drum transcription. *CoRR*, abs/1906.03697, 2019.

[7] C. Dittmar and D. Gärtner. Real-time transcription and separation of drum recordings based on nmf decomposition. In *DAFx*, pages 187–194, 2014.

[8] S. Dixon. On the computer recognition of solo piano music. In *Proceedings of Australasian computer music conference*, pages 31–37, 2000.

[9] G. Dzhambazov. Towards a drum transcription system aware of bar position. In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*. Audio Engineering Society, 2014.

[10] J. Gardner, I. Simon, E. Manilow, C. Hawthorne, and J. Engel. Mt3: Multi-task multitrack music transcription, 2022.

[11] O. Gillet and G. Richard. Automatic transcription of drum loops. In *2004 IEEE International Conference on Acoustics, Speech, and Signal Processing*, volume 4, pages iv–iv, 2004.

[12] C. Hawthorne, E. Elsen, J. Song, A. Roberts, I. Simon, C. Raffel, J. H. Engel, S. Oore, and D. Eck. Onsets and frames: Dual-objective piano transcription. *CoRR*, abs/1710.11153, 2017.

[13] C. Hawthorne, I. Simon, R. Swavely, E. Manilow, and J. Engel. Sequence-to-sequence piano transcription with transformers, 2021.

[14] C. Jacques and A. Roebel. Automatic drum transcription with convolutional neural networks. In *21th International Conference on Digital Audio Effects, Sep 2018, Aveiro, Portugal*, 2018.

[15] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[16] A. Klapuri and M. Davy. *Signal Processing Methods for Music Transcription*. Springer US, 2007.

[17] H. Lindsay-Smith, S. McDonald, and M. Sandler. Drumkit transcription via convolutive nmf. In *International Conference on Digital Audio Effects (DAFx), York, UK*, 2012.

[18] E. Manilow, P. Seetharaman, and B. Pardo. Simultaneous separation and transcription of mixtures with multiple polyphonic and percussive instruments. In *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 771–775, 2020.

[19] M. Miron, M. E. Davies, and F. Gouyon. An open-source drum transcription system for pure data and max msp. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 221–225, 2013.

[20] J. A. Moorer. On the transcription of musical sound by computer. *Computer Music Journal*, 1(4):32–38, 1977.

[21] J. Paulus and A. Klapuri. Drum sound detection in polyphonic music with hidden markov models. *EURASIP Journal on Audio, Speech, and Music Processing*, 2009:1–9, 2009.

[22] M. Piszczalski and B. A. Galler. Automatic music transcription. *Computer Music Journal*, 1(4):24–31, 1977.

[23] G. E. Poliner and D. P. Ellis. A discriminative model for polyphonic piano transcription. *EURASIP Journal on Advances in Signal Processing*, 2007:1–9, 2006.

[24] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, and D. P. W. Ellis. Mir_eval: A transparent implementation of common mir metrics. In *International Society for Music Information Retrieval Conference*, 2014.

[25] C. Raphael. Automatic transcription of piano music. In *ISMIR*, 2002.

[26] A. Roebel, J. Pons, M. Liuni, and M. Lagrangey. On automatic drum transcription using non-negative matrix deconvolution and itakura saito divergence. In

2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 414–418. IEEE, 2015.

[27] M. Rossignol, M. Lagrange, G. Lafay, and E. Benetos. Alternate level clustering for drum transcription. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 2023–2027, 2015.

[28] S. Sigtia, E. Benetos, and S. Dixon. An end-to-end neural network for polyphonic piano music transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 24(5):927–939, 2016.

[29] C. Southall, R. Stables, and J. Hockman. Automatic drum transcription for polyphonic recordings using soft attention mechanisms and convolutional neural networks. 2017.

[30] V. M. A. Souza, G. E. A. P. A. Batista, and N. E. Souza-Filho. Automatic classification of drum sounds with indefinite pitch. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2015.

[31] L. Thompson, M. Mauch, S. Dixon, et al. Drum transcription via classification of bar-level rhythmic patterns. 2014.

[32] K. Toyama, T. Akama, Y. Ikemiya, Y. Takida, W.-H. Liao, and Y. Mitsufuji. Automatic piano transcription with hierarchical frequency-time transformer, 2023.

[33] S. Ueda, K. Shibata, Y. Wada, R. Nishikimi, E. Nakamura, and K. Yoshii. Bayesian drum transcription based on nonnegative matrix factor decomposition with a deep score prior. In *ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 456–460, 2019.

[34] R. Vogl, M. Dorfer, and P. Knees. Recurrent neural networks for drum transcription. In *ISMIR*, pages 730–736, 2016.

[35] R. Vogl, M. Dorfer, and P. Knees. Drum transcription from polyphonic music with recurrent neural networks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 201–205, 2017.

[36] R. Vogl, M. Dorfer, G. Widmer, and P. Knees. Drum transcription via joint beat and drum modeling using convolutional recurrent neural networks. In *ISMIR*, pages 150–157, 2017.

[37] R. Vogl, G. Widmer, and P. Knees. Towards multi-instrument drum transcription. *CoRR*, abs/1806.06676, 2018.

[38] I.-C. Wei, C.-W. Wu, and L. Su. Improving automatic drum transcription using large-scale audio-to-midi aligned data. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 246–250, 2021.

[39] C.-W. Wu, C. Dittmar, C. Southall, R. Vogl, G. Widmer, J. Hockman, M. Müller, and A. Lerch. A review of automatic drum transcription. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 26(9):1457–1483, 2018.

[40] C.-W. Wu and A. Lerch. Drum transcription using partially fixed non-negative matrix factorization. In *2015 23rd European Signal Processing Conference (EUSIPCO)*, pages 1281–1285, 2015.

[41] Y.-T. Wu, Y.-J. Luo, T.-P. Chen, I.-C. Wei, J.-Y. Hsu, Y.-C. Chuang, and L. Su. Omnizart: A general toolbox for automatic music transcription, 2021.

[42] T. Xu, P. Dai, B. He, M. Zhang, and J. Zhu. Automatic drum transcription with label augmentation using convolutional neural networks. In T. Mantoro, M. Lee, M. A. Ayu, K. W. Wong, and A. N. Hidayanto, editors, *Neural Information Processing*, pages 65–76, Cham, 2021. Springer International Publishing.

[43] Y.-Y. Yang, M. Hira, Z. Ni, A. Chourdia, A. Astafurov, C. Chen, C.-F. Yeh, C. Puhrsch, D. Pollack, D. Genzel, D. Greenberg, E. Z. Yang, J. Lian, J. Mahadeokar, J. Hwang, J. Chen, P. Goldsborough, P. Roy, S. Narenthiran, S. Watanabe, S. Chintala, V. Quenneville-Bélair, and Y. Shi. Torchaudio: Building blocks for audio and speech processing, 2022.