

JOINT TRANSCRIPTION OF ACOUSTIC GUITAR STRUMMING DIRECTIONS AND CHORDS

Sebastian Murgul^{1,2}

Johannes Schimper²

Michael Heizmann²

¹ Klangio GmbH, Karlsruhe, Germany

² Karlsruhe Institute of Technology, Karlsruhe, Germany

sebastian.murgul@klang.io

ABSTRACT

Automatic transcription of guitar strumming is an under-represented and challenging task in Music Information Retrieval (MIR), particularly for extracting both strumming directions and chord progressions from audio signals. While existing methods show promise, their effectiveness is often hindered by limited datasets. In this work, we extend a multimodal approach to guitar strumming transcription by introducing a novel dataset and a deep learning-based transcription model. We collect 90 min of real-world guitar recordings using an ESP32 smartwatch motion sensor and a structured recording protocol, complemented by a synthetic dataset of 4 h of labeled strumming audio. A Convolutional Recurrent Neural Network (CRNN) model is trained to detect strumming events, classify their direction, and identify the corresponding chords using only microphone audio. Our evaluation demonstrates significant improvements over baseline onset detection algorithms, with a hybrid method combining synthetic and real-world data achieving the highest accuracy for both strumming action detection and chord classification. These results highlight the potential of deep learning for robust guitar strumming transcription and open new avenues for automatic rhythm guitar analysis.

1. INTRODUCTION

Automatic music transcription is a key task in Music Information Retrieval (MIR), aiming to convert audio signals into symbolic representations. For the transcription of solo instrument music, numerous new approaches and tools have been proposed over the last years [1]. While classical note-tracking models such as [2], [3], and [4] perform well for fingerpicking, they are not designed to predict strumming directions. These models focus on individual note onsets and often struggle with the dense polyphony and rhythmic structure of strumming, where the emphasis lies on chord-level articulation. This limitation highlights the need for a dedicated strumming transcription system with applications in music education, DAW plugins, and notation software.

Research on guitar strumming transcription has primarily followed two main approaches: audio-based classification and sensor-based motion analysis. In 2019, Bello et al. proposed a neural network-based classification system to distinguish between up and down strokes using Mel-Frequency Cepstral Coefficients (MFCCs) segments as input features [5]. Their approach achieved a classification accuracy of 72.5 % for a Convolutional Neural Network (CNN) and 70 % for a Long Short-Term Memory (LSTM) model. Earlier, in 2013, Matsushita et al. developed a wristwatch-like device designed to analyze down-strumming actions in terms of note timing and intensity [6]. More recently, Freire et al. (2020) explored strumming gestures in greater detail using inertial measurement units (IMUs) and motion capture technology, further advancing sensor-based analysis of guitar performance [7]. A multimodal approach was introduced in 2022 by Murgul et al., who combined a back-of-hand-mounted motion sensor with guitar pickup audio for strumming action transcription [8]. Their method involved recording a small manually labeled dataset, which was used to evaluate algorithmic annotation techniques based on onset detection in the pickup signal and thresholding the first-order derivative of the motion data.

Building on the approach in Murgul et al., we extend the multimodal approach to create a bigger and more diverse dataset in order to train a neural network. We increase the dataset size from 5 min to 90 min and from 4 chords to 24 chords (major / minor) while also adding more complex strumming rhythms and performance parameter variations. Therefore, an improved hand motion sensor based on an off-the-shelf ESP32 smartwatch module is developed, and a sophisticated recording plan with specific instructions to the players is created. A new guitar strumming dataset is recorded by three guitar players using this approach and semi-automatically annotated using the multimodal information. While the semi-automatic annotation process is scalable, the recording process still does take some time. Therefore, to complement the real-world dataset, we present a guitar strumming data synthesis approach that is used to generate an additional 4 h of labeled strumming audio. These datasets are then used to train a CRNN model to automatically detect strumming events and classify the strumming direction as well as the played chord from solely microphone audio. Finally, the transcription results are evaluated using the test split of the real strumming recordings and compared with baseline algorithms.



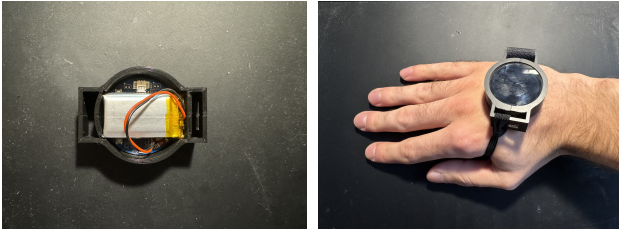
© S. Murgul, J. Schimper, and M. Heizmann. Licensed under a Creative Commons Attribution 4.0 International License (CC BY 4.0). **Attribution:** S. Murgul, J. Schimper, and M. Heizmann, “Joint Transcription of Acoustic Guitar Strumming Directions and Chords”, in *Proc. of the 26th Int. Society for Music Information Retrieval Conf.*, Daejeon, South Korea, 2025.

2. MULTIMODAL STRUMMING RECORDING

2.1 Motion Recording Hardware

To capture hand movement and, consequently, the strumming direction, a compact and lightweight system is required that can be attached to the playing hand. It must enable wireless communication for transmitting motion data and be capable of starting and stopping audio recordings on a computer via wireless commands. Additionally, the system should be intuitive for guitarists to use. For scalable applications, the solution should be cost-efficient. The ESP32-S3-Touch-LCD-1.28 module from Waveshare meets these requirements and serves as the central micro-controller [9]. It features a 3-axis accelerometer (QMI8658), a LiPo battery connector with a battery management, and supports the wireless standards Wi-Fi and Bluetooth Low Energy (BLE). Furthermore, the module includes an LCD screen with touch functionality and a compact form factor.

A custom 3D-printed enclosure enables a watch-like attachment on the back of the hand. The enclosure also houses a 350mAh LiPo battery, as shown in Figure 1a. Figure 1b illustrates the sensor system attached to the back of the hand.



(a) Backside of the hand sensor without cover. (b) Sensor attached to the back of the hand.

Figure 1. Hand sensor in its enclosure

Hand movement is described using a simplified model like in [8], in which the hand performs a semicircular motion around the elbow. The x -axis runs along the back of the hand, orthogonal to the fingers, while the y -axis is orthogonal to the x -direction, pointing towards the fingertips. The relevant acceleration components are gravitational acceleration A_g , centripetal acceleration $A_{\text{centripetal}}$, and tangential acceleration $A_{\text{tangential}}$ [10]. The spatial orientations recorded by the sensor, along with the measured accelerations for different hand positions, are shown in Figure 2. The centripetal acceleration acts exclusively in the y -direction, while the tangential acceleration occurs along the x -axis. Consequently, the acceleration A_x in the x -direction and A_y in the y -direction are given by

$$A_x = A_{\text{tangential}} + A_g \cdot \cos(\phi) \quad (1)$$

$$A_y = A_{\text{centripetal}} + A_g \cdot \sin(\phi) \quad (2)$$

where ϕ is the angle relative to the horizontal axis, ranging from -90° to 90° . For slow, quasi-stationary hand movements, A_x ranges from $-1g$ to $0g$, while A_y takes values between $-1g$ and $1g$. Due to the symmetry properties of the sine function, A_x alone cannot determine the movement

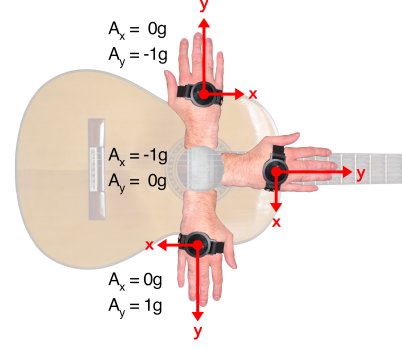


Figure 2. Motion model of the sensor

direction. However, by differentiating the acceleration in the y -direction, the movement direction can be inferred. A negative gradient corresponds to an upward motion, while a positive gradient corresponds to a downward motion.

In non-stationary cases, such as during strumming, both tangential and centripetal acceleration contribute to A_x and A_y respectively alongside the gravitational acceleration. The y -direction experiences an additional, constant centripetal acceleration. Because our method relies on acceleration derivatives, the constant centripetal acceleration can be ignored.

2.2 Recording Process

Table 1 gives an overview of the playing instructions given to the guitarists. To compile the datasets, 28 different strumming patterns in 4/4 time signature based on [11, 12] are used, ranging from rhythmically simple to complex syncopated patterns. The patterns vary in parameters like tempo (60, 80, 100 BPM), chord progressions, playing style (plectrum, finger), and volume (soft, medium, loud). The variations were determined randomly based on a uniform distribution.

Parameter	Values
Pattern	28 patterns
Tempo	60 BPM, 80 BPM, 100 BPM
Movement	little, normal, large
Volume	quiet, medium, loud
Technique	finger, pick
Chords	major and minor chord progressions

Table 1. Results on microphone audio.

The data collection was conducted with three guitarists, including a professional guitar teacher and two experienced amateur guitarists. The strumming patterns were played for 60s each to a metronome, following the predefined parameters. Simultaneously, audio recordings from the guitar pickup and acceleration data were captured. Synchronization of both audio signals was performed using cross-correlation. Additionally, the guitarists' playing was recorded using the microphone on an iPhone 15 Pro. The total recording duration amounts to 90 min. Due to the lightweight design and the mounting position on the back

of the hand, the guitar players’ fingers were not constrained by the sensor during performance.

2.3 Semi-Automatic Annotation

The annotation process involves identifying the onset times and strumming directions within the pickup recordings as well as the synchronization with the motion sensor signal. Instead of relying solely on automated onset detection, the process is optimized by incorporating prior knowledge from the recording plan, which includes tempo, rhythm patterns, chords, and strumming sequences. This structured information allows for a more robust prediction of expected onset times, reducing reliance on purely signal-based onset detection. To determine actual onset times, spectral flux analysis [13] is used to detect significant changes in the audio signal. However, since the guitarist does not necessarily start at the exact zero-second mark, a user-assisted graphical interface is employed to align the estimated onsets with the theoretical pattern. The process involves selecting the actual start time and iteratively adjusting until the detected onsets align with the expected timing based on the metronome. Strumming direction is determined using acceleration data, which is synchronized with the audio signal. Since transmission latency and system delays introduce a time offset between the audio and acceleration data, manual adjustments are required. An interactive visualization displays both spectral flux and differentiated acceleration, allowing users to shift the acceleration data until the peaks of acceleration derivatives align with the detected onsets.

To assign strumming direction, peaks in the acceleration derivative corresponding to upward and downward hand movements are matched with detected onset peaks in spectral flux. If the acceleration derivative is positive at an onset time, it is labeled as an up strum; if negative, it is labeled as a down strum. Next, we use the a priori information from the recording plan to automatically correct the annotations and add chord labels. Since we use a metronome, it can be assumed that the rhythmical pattern is played consistently enough to interpolate missed strumming events. Finally, the annotated data undergoes manual validation and correction by a human annotator. The annotator visually inspects and adjusts the detected onsets and strumming directions using an interactive graphical interface.

3. GUITAR STRUMMING SYNTHESIS

To create a diverse and scalable dataset for training strumming transcription models, we introduce a novel strumming synthesis approach consisting of three stages: strumming tablature sampling, audio rendering, and audio augmentation. This method generates approximately 1000 examples totaling 4 h of audio, which are randomly split into 90 % training, 5 % validation, and 5 % testing sets.

3.1 Strumming Tablature Sampling

The first step involves generating synthetic strumming tablatures, as illustrated in Figure 3. A database of 51 chord

progressions in functional notation and 36 strumming patterns defined on a 16th-note grid serve as the foundation for generating variations. Each example is created by randomly selecting a chord progression, transposing it to a random key, and mapping each chord to a fingering from a lookup table. A random strumming pattern and tempo are then applied to create a complete tablature. To introduce natural imperfections, the last note of a strumming chord is randomly dropped in 50 % of cases, simulating playing inconsistencies typical of amateur guitarists. The generated tablatures are stored in the GuitarPro¹ format, alongside a CSV annotation file containing timing, strumming action, and chord labels.

3.2 Audio Rendering

The synthesized tablatures are rendered into audio using DAWDREAMER [14] and Ample Sound’s virtual guitar instruments², following a methodology similar to SynthTab [15]. Instead of converting tablatures to MIDI, we use *.fxp* preset files to load the GuitarPro notation directly into the virtual instrument engine. This way, up and down stroke information can be input from the tablature. To enhance realism, rendering parameters are randomized, including the blend between virtual microphones and the amount of fret noise introduced. The final output is saved as a 44.1 kHz WAV file. Since the rendering process introduces an average 40 ms latency, this delay is accounted for in the dataset annotations to maintain synchronization accuracy.

3.3 Audio Augmentation

To further improve realism and variability, a post-processing step applies a chain of effects using the Pedalboard library [16]. The augmentation pipeline introduces controlled distortions and environmental factors to better simulate real-world recordings. The processing chain includes distortion, high- and low-pass filtering, and compression to mimic tonal variations across different recording conditions. To simulate room acoustics, a convolutional reverb effect is applied. Additional background noise layers, including ambient recordings (traffic, weather, and living room sounds) and white noise, are incorporated to model microphone imperfections and noisy environments. Finally, short bursts of fretting sounds and percussive noises, such as light tapping or clapping, are injected at random intervals to emulate natural guitar handling. The effect parameters, such as signal-to-noise ratio (SNR), filter cut-off frequencies, and dry/wet mix ratios, are randomized to ensure broad generalization.

4. MODEL

Our model builds upon the Convolutional Recurrent Neural Network (CRNN) architecture proposed by Kong et al. [17] for piano transcription. Unlike traditional classification-based approaches that estimate a discrete piano roll representation, this method employs a regression-based strategy

¹ See <https://www.guitar-pro.com> for more information.

² Available at <https://amplesound.net/en/index.asp>.

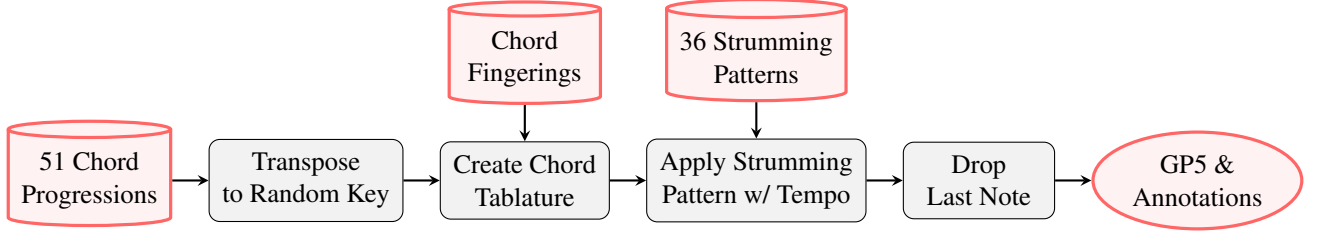


Figure 3. Flow chart of the strumming tablature sampling process.

to predict the time to the next onset or offset event. This design allows for more precise onset estimations beyond the limitations of fixed frame step sizes, while also increasing robustness against minor misalignments in onset label annotations during training.

4.1 Pre-Processing

The input audio is resampled to 16 kHz and segmented into overlapping 10 s clips with a hop size of 1 s to enhance data diversity. Each segment is converted into a logarithmic Mel spectrogram, which serves as the input representation for the neural network. The spectrogram is computed using a window size of 2048 samples and a hop size of 160 samples, resulting in a time-frequency representation with 229 frequency bins, starting at a minimum frequency of 30 Hz. To improve generalization, random pitch shifts in the range $[-6, 6]$ semitones are applied during training, with chord labels transposed accordingly. The overlapping segmentation and augmentation ensure robust feature learning across diverse strumming patterns.

4.2 Architecture

The model consists of two main components: a strumming onset regression network and a chord classification network. The input Mel spectrogram is first processed by a convolutional layer stack (Conv Stack) designed to capture time-frequency features. The structure of the Conv Stack follows the design in [17] and consists of four convolutional blocks. Each block contains two convolutional layers with identical kernel sizes, followed by a pooling operation that reduces the spectral dimension while preserving temporal information. After the final convolutional block, the extracted features are flattened for subsequent processing. The flattened feature representation is passed through a fully connected (FC) layer before being fed into a bidirectional GRU (biGRU) layer with 256 units. The output of the biGRU is then passed through another fully connected layer, which generates regression values for up strums and down strums.

In parallel to the onset regression, a separate chord feature extraction stack processes the input spectrogram in a similar manner. Since chord labels are only available at strumming event times, the outputs of both networks are merged before passing through an additional biGRU and fully connected layer to produce final classification logits

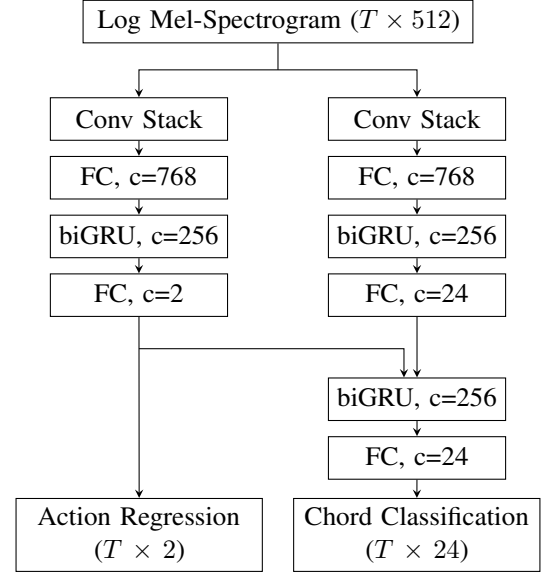


Figure 4. Joint strumming action detection and chord recognition network using logarithmic Mel spectrogram as input feature.

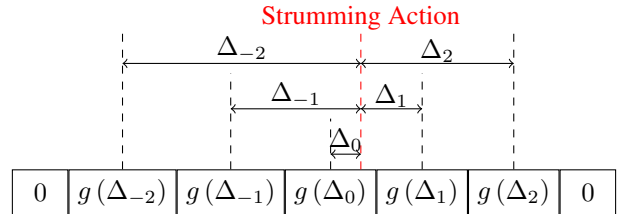


Figure 5. Structure of Strumming Action Onset Regression Labels.

for 24 major and minor chord classes. Figure 4 provides an overview of the full model architecture.

4.3 Regression Targets

Instead of relying on binary frame-based labels, a regression-based approach is used to determine strumming actions, as illustrated in Figure 5. The regression target function $g(\Delta_i) \in [0, 1]$ encodes the time difference to the next strumming action onset Δ_i , where i is the index of a frame, using a triangular distribution. The target is defined as

$$g(\Delta_i) = \begin{cases} 1 - \frac{|\Delta_i|}{J\Delta}, & |i| \leq J \\ 0, & |i| > J \end{cases}, \quad (3)$$

where Δ denotes the frame hop size and J is a hyperparameter that controls the sharpness of the regression labels which is set to 5 in our experiments. The loss function consists of two components: one for strumming onset regression and another for chord classification. The strumming action regression loss l_{action} is calculated from the regression output R_{action} and the target G_{action} by

$$l_{\text{action}} = \sum_{t=1}^T \sum_{k=1}^K l_{\text{bce}}(G_{\text{action}}(t, k), R_{\text{action}}(t, k)) \quad , \quad (4)$$

where l_{bce} represents the binary cross-entropy loss, T is the number of time steps, and K denotes the number of strumming action categories. For chord classification, a similar loss function is used on the prediction outputs P_{chord} and the targets G_{chord} :

$$l_{\text{chord}} = \sum_{t=1}^T \sum_{c=1}^C l_{\text{bce}}(G_{\text{chord}}(t, c), P_{\text{chord}}(t, c)) \quad . \quad (5)$$

where C represents the number of possible chord labels. The total loss function used during training is simply the sum of both components:

$$l = l_{\text{action}} + l_{\text{chord}} \quad . \quad (6)$$

The model is trained using the AdamW optimizer [18] with an initial learning rate of 10^{-4} . The training process is run for 20,000 steps with a batch size of 6. On an NVIDIA Tesla V100 GPU, training takes approximately 2 h.

5. EXPERIMENTS AND RESULTS

This section evaluates the performance of our proposed method for strumming onset detection, direction classification, and chord recognition. We begin by assessing the detection accuracy using guitar pickup signals, followed by an evaluation of real-world microphone recordings. Finally, we analyze the effectiveness of pitch shift augmentation and compare our chord recognition with existing approaches.

Model performance is measured using precision, recall, and F1-score for strumming detection. Specifically, we report these metrics for down strums ($F1_{\text{down}}$), up strums ($F1_{\text{up}}$), and strumming class agnostic ($F1_{\text{any}}$). A 50 ms tolerance window is used, following the `mir_eval` library [19].

5.1 Results on Guitar Pickup Signals

In our first experiment, we explore the performance of our model directly on the guitar pickup signals. We use two of the guitarists we recorded to train our model and evaluate on the third guitarist. We compare the detection quality of our trained model with common onset detection functions spectral flux [13], super flux [13] and Complex Domain Onset Detection Function (CD-ODF) [20]. For spectral flux and super flux, we use the implementation given in the `librosa` library [21]. The resulting precision, recall, and F1-score for any strumming direction are highlighted in Table 2 for comparison. Of the onset detection functions, the spectral flux offers the best detection results, directly followed by

the CD-ODF. Compared with spectral flux and super flux, the CD-ODF offers a noticeably high recall. Therefore, it might be suitable for an active learning labeling scenario. Our model outperforms the onset detection functions in all three precision, recall and F1-score. By achieving an F1-score of about 98 %, the model is quite capable of reliably detecting the strumming actions in the pickup signal.

Method	F1 _{any}	P _{any}	R _{any}
Spectral Flux [13]	79.49 %	78.53 %	81.86 %
Super Flux [13]	74.36 %	77.04 %	73.36 %
CD-ODF [20]	79.32 %	68.50 %	98.15 %
Ours	97.60 %	96.54 %	98.73 %

Table 2. Strumming detection results on pickup audio.

By matching the detected strumming onsets with the movement data from the hand sensor, the strumming direction can also be determined. In Table 3, we compare the results of the multimodal algorithmic approach with our CRNN model. For all four approaches, the F1-score for down strums is higher than for up strums. Our CRNN model outperforms the algorithmic approaches for the down strum as well as the up strum class, whereby the increase is specifically noticeable for up strum events. Combining the CRNN detection with the acceleration-based classification leads to the overall best results. Therefore, the labeling could be automated quite efficiently by using a hybrid approach with the pickup audio signal to detect the events in the audio and the motion sensor data to get the strumming event class algorithmically.

Methods	F1 _{any}	F1 _{down}	F1 _{up}
Spectral Flux [13]	79.49 %	85.40 %	68.60 %
Super Flux [13]	74.36 %	84.40 %	67.80 %
CD-ODF [20]	79.32 %	82.20 %	78.40 %
Ours	97.60 %	87.87 %	84.90 %
Ours + Sensor	97.60 %	90.02 %	88.66 %

Table 3. Strumming event detection results by class. The onset detection function results are paired with the hand movement signal in order to classify the events.

5.2 Results on Microphone Recordings

Next, we examine the action detection performance on the real-world microphone data. The real-world audio contains overall more noise, reverb and ambient sounds. The detection performance for different training dataset constellations (Synthetic (Sy), microphone exclusively (Ph), microphone and pickup (Ph + Pi), and all three datasets (Sy + Ph + Pi)) is compared in Table 4. The $F1_{\text{any}}$ results for all datasets lie in a similar range. The synthetic dataset achieves about 5 % better results than when only using the comparably small training dataset of real-world phone recordings. When the pickup audio dataset is used in addition to the microphone recordings, we see a clear increase across all models. The

Training Data	$F1_{any}$	R_{any}	P_{any}	$F1_{down}$	R_{down}	P_{down}	$F1_{up}$	R_{up}	P_{up}
Sy	89.77 %	89.47 %	90.56 %	73.92 %	75.00 %	74.04 %	52.64 %	56.99 %	51.04 %
Ph	85.06 %	84.11 %	86.12 %	79.90 %	78.70 %	81.42 %	66.81 %	67.52 %	67.88 %
Ph + Pi	89.45 %	88.37 %	90.64 %	82.94 %	83.72 %	82.40 %	75.10 %	73.17 %	78.24 %
Sy + Ph + Pi	92.75 %	92.50 %	93.25 %	85.51 %	85.87 %	85.43 %	79.02 %	81.15 %	77.80 %

Table 4. Results on microphone audio trained on various combinations of the synthetic dataset (Sy), real-world pickup audio (Pi), and real-world microphone recordings (Ph).

increase is especially significant for up strums. In general, the real-world data performs significantly better than the synthetic dataset exclusively. Here, we see an increase of over 40 % compared to the synthetic dataset exclusively. Therefore, reliable onset detection itself can be trained from synthetic examples alone, but the classification of the strumming action profits from real-world audio. The best overall results are obtained by combining the synthetic dataset with the microphone and pickup dataset. This indicates that increasing the real-world dataset in additional recording sessions might yield further improvements. Interestingly, fine-tuning a checkpoint pretrained on synthetic data on the phone and pickup data leads to worse results than joining all three training datasets.

5.3 Effect of Pitch Shift Augmentation

Max Pitch Shift	$F1_{any}$	$F1_{down}$	$F1_{up}$
None	81.15 %	71.04 %	55.80 %
± 3 semitones	85.06 %	79.10 %	71.99 %
± 6 semitones	89.45 %	82.94 %	75.10 %
± 12 semitones	85.90 %	80.89 %	72.25 %

Table 5. Effect of the max pitch shift parameter in the pre-processing step on the strumming detection performance.

In the model pre-processing we perform data augmentation in the form of a random pitch shift before calculating the input spectrogram. The effect of the pitch shift augmentation is studied using the training on the combined phone and pickup dataset. The results of this experiment are shown in Table 5. Applying a max pitch shift of 6 semitones leads to the best results. The F1-score for down strums increases by 10 % and up strums F1-score by 14 %. While the pitch shift introduces more artifacts as the note shift increases, it also increases the diversity of chords used and therefore helps the model generalize.

5.4 Chord Recognition

While the previous experiments only focused on the strumming action detection and classification, the chord recognition performance is quantified in this experiment and compared with a popular CNN-based [22] and a state-of-the art transformer model [23]. We use the checkpoints provided by the authors. In contrast to the chord recognition task, where typically a musical piece is segmented into sections of a specific chord, we are interested in assigning a chord to

Method (Dataset)	Accuracy
Deep Chroma Chord Recognition [22]	80.37 %
Chord Recognition BTC [23]	89.21 %
Ours (Sy)	87.84 %
Ours (Ph + Pi)	81.52 %
Ours (Sy + Ph + Pi)	90.06 %

Table 6. Results for chord recognition on the microphone audio of the real-world recordings.

a detected strumming event. Therefore, we use the ground truth strumming action times to determine a chord label. For the training of our own model, we use a maximum pitch shift of 6 semitones. The resulting accuracy scores for the major-minor vocabulary are shown in Table 6. The chord recognition transformer model and our model trained on the combined dataset achieve the best results of about 90 %. The CNN-based chord tracking shows the weakest performance. In contrast to the strumming action detection, our model trained on the synthetic dataset alone performs significantly better than with only the smaller real-world dataset. Training on all three datasets further increases the performance of our approach.

6. CONCLUSION

This study demonstrates the effectiveness of a CRNN-based model for the joint transcription of guitar strumming actions and chords. We introduced a novel approach to strumming synthesis, generating a large dataset of synthetic strumming examples. By extending an existing multimodal strumming transcription framework, we also collected 90 minutes of real-world guitar recordings, enhanced with semi-automatic annotations. The combination of synthetic and real-world datasets allowed us to train a robust transcription model capable of accurately detecting strumming onsets, classifying strumming direction, and identifying chords from microphone audio.

Future work could extend this approach to cover a broader range of rhythmic patterns, including muted strumming events, which pose a challenge for motion-based annotation methods. Additionally, the chord vocabulary, currently limited to major and minor chords, could be expanded to include seventh chords, suspended chords, and other common chord voicings. These improvements would further enhance the versatility and real-world applicability of automatic strumming transcription models.

7. REFERENCES

- [1] E. Benetos, S. Dixon, Z. Duan, and S. Ewert, "Automatic Music Transcription: An Overview," *IEEE Signal Processing Magazine*, vol. 36, no. 1, pp. 20–30, 2018.
- [2] X. Riley, D. Edwards, and S. Dixon, "High Resolution Guitar Transcription via Domain Adaptation," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 1051–1055.
- [3] S. Chang, E. Benetos, H. Kirchhoff, and S. Dixon, "YourMT3+: Multi-Instrument Music Transcription with Enhanced Transformer Architectures and Cross-Dataset STEM Augmentation," in *2024 IEEE 34th International Workshop on Machine Learning for Signal Processing (MLSP)*, 2024, pp. 1–6.
- [4] A. Wiggins and Y. Kim, "Guitar Tablature Estimation With a Convolutional Neural Network," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019, pp. 284–291.
- [5] K. Bello and P. Mayol, "Classification of Acoustic Guitar Strum using Convolutional Neural Networks and Long-Short-Term-Memory," *Philippine e-Journal for Applied Research and Development*, vol. 9, pp. 49–57, 2019.
- [6] S. Matsushita and D. Iwase, "Detecting Strumming Action While Playing Guitar," in *Proceedings of the 2013 International Symposium on Wearable Computers*, 2013, pp. 145–146.
- [7] S. Freire, G. Santos, A. Armondes, E. Meneses, and M. Wanderley, "Evaluation of Inertial Sensor Data by a Comparison With Optical Motion Capture Data of Guitar Strumming Gestures," *Sensors*, vol. 20, no. 19, p. 5722, 2020.
- [8] S. Murgul and M. Heizmann, "A Multimodal Approach to Acoustic Guitar Strumming Action Transcription," in *Extended Abstracts for the Late-Breaking Demo Session of the 23rd International Society for Music Information Retrieval Conference (ISMIR)*, 2022.
- [9] Waveshare. (2025) Esp32-s3 touch lcd 1.28". <https://www.waveshare.com/esp32-s3-touch-lcd-1.28.htm>. (accessed Feb. 28, 2025).
- [10] D. Kleppner and R. J. Kolenkow, *An Introduction To Mechanics*, 2nd ed. Cambridge, UK: Cambridge University Press, 2014.
- [11] D. Samra. (2025) Schlagmuster für Gitarre. <https://www.gitarrenpark.de/blog/schlagmuster-gitarre-strumming-patterns/>. (accessed Feb. 28, 2025).
- [12] E. Swanson. (2025) Strumming Patterns. <https://www.eriksguitarlessons.com/wp-content/uploads/2015/02/Strumming-Patterns-for-Guitar1.pdf>. (accessed Feb. 28, 2025).
- [13] S. Böck and G. Widmer, "Maximum Filter Vibrato Suppression for Onset Detection," in *Proceedings of the 16th International Conference on Digital Audio Effects (DAFx)*, 2013, p. 4.
- [14] D. Braun, "DawDreamer: Bridging the Gap Between Digital Audio Workstations and Python Interfaces," in *Extended Abstracts for the Late-Breaking Demo Session of the 22nd International Society for Music Information Retrieval Conference (ISMIR)*, 2021.
- [15] Y. Zang, Y. Zhong, F. Cwitkowitz, and Z. Duan, "Synthtab: Leveraging Synthesized Data for Guitar Tablature Transcription," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2024, pp. 1286–1290.
- [16] P. Sobot, "Pedalboard," Jul. 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.7817838>
- [17] Q. Kong, B. Li, X. Song, Y. Wan, and Y. Wang, "High-Resolution Piano Transcription With Pedals by Regressing Onset and Offset Times," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 3707–3717, 2021.
- [18] I. Loshchilov and F. Hutter, "Decoupled Weight Decay Regularization," in *International Conference on Learning Representations (ICLR)*, 2017.
- [19] C. Raffel, B. McFee, E. J. Humphrey, J. Salamon, O. Nieto, D. Liang, D. P. Ellis, and C. C. Raffel, "MIR_EVAL: A Transparent Implementation of Common MIR Metrics," in *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR)*, 2014, p. 2014.
- [20] J. P. Bello, C. Duxbury, M. Davies, and M. Sandler, "On the Use of Phase and Energy for Musical Onset Detection in the Complex Domain," *IEEE Signal Processing Letters*, vol. 11, no. 6, pp. 553–556, May 2004.
- [21] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "Librosa: Audio and Music Signal Analysis in Python," *SciPy*, vol. 2015, pp. 18–24, 2015.
- [22] F. Korzeniowski and G. Widmer, "Feature Learning for Chord Recognition: The Deep Chroma Extractor," in *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR)*, 2016.
- [23] J. Park, K. Choi, S. Jeon, D. Kim, and J. Park, "A Bi-Directional Transformer for Musical Chord Recognition," in *Proceedings of the 20th International Society for Music Information Retrieval Conference (ISMIR)*, 2019.