# Automatic Guitar Tablature Generation for Monophonic Melodies

**Kezia Devathasan**
**V00888007**
**keziadevathasan@uvic.ca**

**Elliot Lupini**
**V00775566**
**elupini@uvic.ca**

**ABSTRACT**

The final iteration of our project report contains a summary of our progress for an automatic guitar tablature generation model. While the motivation behind our project remains relatively static, we discuss what has worked well, the major challenges faced, and describe how far we were able to get with our project. We also reflect on what we have learned in the process and how scholarly works in the field of music information retrieval have guided our work.

## 1. INTRODUCTION

Our project continued to focus on the concept of automatic guitar tablature (tab) generation. We specifically focused on tab generation for simple, monophonic melodies. Alongside the material we have learned in CSC475, we employed previous knowledge of machine learning algorithms to train and test our program, in the hopes that our program would yield a fair accuracy by the time this project is complete. In this section we discuss our motivations for planning such a project, as well as reflect on a few initially anticipated challenges.

### 1.1 Motivation

Our project idea had several motivating factors. Firstly, our program was intended for guitarists who could not write their own tabs. There might be many reasons for not being able to write one's own tabs, including inexperience and disability. Baker points out that in visually impaired musicians, the inability to access sheet music was a lifelong perceived barrier to learning [5]. Although our program was not designed with specific disabilities in mind, we saw use for a successful project within this area. Secondly, writing tabs by hand is very time consuming; a program that can automate this process will save musicians and composers time, and if well executed, will assist in mitigating human error. Lastly, tabs make playing simple songs much more accessible to beginner musicians [20] than standard notation because tabs are fairly simple to learn. We envisioned a final project of this nature to be used by beginner musicians who perhaps want to share their own compositions.

### 1.2 Foreseeable Challenges

One of the main challenges we were initially concerned about was the redundancy of guitar notes—that is, the ability to play the same pitch in a variety of ways depending on the combination of fret and string. This challenge could be mitigated by generating standard notation instead of guitar tabs, however standard notation tends to be more difficult for beginner musicians to play. Albasca *et al.* also note that there is an abundance of software to generate notation for other instruments, but such products are lacking for guitar players [15].

Furthermore, Albasca *et al.* noted that pitch detection was a significant issue in their work [15]. However, we recognize that this study does not have the same constraints, as the authors allowed for polyphonic elements in their melodies which appears to significantly complicate the process of detecting pitch. We believed that by having added the constraint of only allowing monophonic signals, we had reduced the difficulty of our pitch detection tasks. An explanation of a shift away from this task follows in this project report.

Initially, we were also concerned about being able to generate any tabs at all, as neither of us had created anything like this. However, this was not as significant of a challenge as we initially anticipated. Initially, were able to create helper functions that ultimately generated tabs based on a training set input. These tabs were not *good* tabs; that is, they were not particularly playable, and they were not necessarily displaying the best fingering sequences, but we were able to get to the point of simply visualizing and generating a tab

(Figure 1) fairly early in the process which was encouraging.

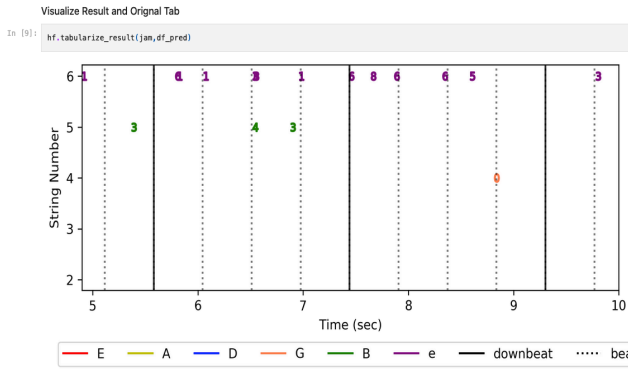


**Figure 1:** early-stage tablature generated with implemented helper functions and training data set.

Another significant challenge posed was our inexperience with such a project. We had many ideas about what the implementation for our project may look like. After spending ample time researching previous related works we realized we had some issues with the scope of our project, and refocused. These challenges are outlined in Section 2.2.

## 2. PROJECT METHODS

In the following section we discuss the implementation we used for our project and subsequent outcome (Section 2.1), and the major obstacles we faced along the way (Section 2.2). We also discuss the related works that guided us towards our final product in Section 2.3. Finally, in Section 2.4, we reflect on our roles in the project, and what we have learned in the process.

### 2.1 Project Implementation and Outcomes

The final goal of the project was to design a program that takes as input a series of monophonic signals, and outputs a guitar tab representative of the input signals. We hoped to accomplish this using deep learning, where the input into the model is the melody, and the label is the guitar tab. We intended to use descriptive, pre-labeled data, which meant that we would use an existing pitch detection implementation to be able to pass and train the model to create the correct information to generate the tabs.

We started by creating an algorithm to predict the string and fret positions based off of the frequency, time, and duration of a note. The assumption that was initially made was that the monophonic input had gone through a pitch detection algorithm already, and that the beat information had been preset by the user. This initial model simply picked the lowest possible fret position for every note. *(By lowest possible fret, we mean where the fret is closest to the nut).* While this was somewhat ridiculous at first, we needed to establish some sort of baseline measure before we could implement more complex algorithms. It also initially served as a placeholder when setting up the pipeline needed to process all the data and output a visualization of a tablature. This early visualization is shown in Figure 1.

After establishing this baseline, we moved onto a Naïve Shortest Path model that would choose an initial voicing that was closest to the nut and then pick the next note that minimizes some cost function. The cost function is based on Manhattan distance, where the strings are the horizontal distance, and the frets are the vertical distance.

Next, we implemented an improved shortest distance algorithm to try and reduce the distance on the tab between each note to optimize playability. This algorithm follows the following logic:

1. Choose *n* initial voicings that maximize distance on the fretboard
2. For each of *n* voicings, compute the cost of *m* possible next options.
3. Select *x* lowest cost options and add to current cost for each path
4. Repeat steps 2, 3.

When creating the shortest distance algorithm, we realized that we needed to create multiple possible initializations. If we did not do this, we ended up in a very similar situation we were in earlier with the naïve shortest path algorithm, where a bad initialization (perhaps as bad as the lowest possible fret position) could negatively impact the entire tab prediction. We also observed that expanding the number of paths considered at any given time significantly helped, however only up until a certain number of paths. When we exceeded 15 possible paths, the improvement in the accuracy plateaued.

Although we are not completely sure of why this is, (15 possible paths seems rather arbitrary), we considered this a success, as there were significant improvements in the distance between tabs, thus resulting in much more playable tabs. As a short aside,

we considered that measuring accuracy in this sense is somewhat subjective. Since there are multiple ways to play the same tab on a guitar, we would likely need to have an experienced guitarist try out our tabs to determine what is playable; their opinions are likely to differ from our expectations as inexperienced players.

Finally, after implementing a baseline, naïve shortest path, and improved shortest path approach, we implemented a machine learning approach using the MLP classifier from scikit-learn. As input, this algorithm took a midi note, start time, and duration of the note. The machine learning model trains on a specified number of tracks and then predicts the rest of them. It then adds another track from the test set to training set and performs the same prediction. This method of varying the size of the train and test set did not affect the testing accuracy. A 50/50 split and 80/20 split yielded the same accuracies. Further, the test accuracy was consistently higher than the training accuracy, which is not very common. This suggests our model was underfitting.

There are likely two main reasons for this underfitting. Firstly, there is likely not enough training data. Secondly, the model is probably still too simplistic. We performed a grid search to find the best parameters, but the search was rather limited due to computational resources, and did not explore too many variations of hidden layers and sizes. A more thorough search would hopefully help combat the underfitting that is occurring. Additional data would also be very beneficial.

In terms of training and testing datasets, we used the open source dataset of guitar recordings created by Xi *et al.* as part of a project at NYU's Music and Audio Research lab [18]. The guitar recordings in this dataset came with information about tempo, fret positions etc. which meant that we were be able to determine the accuracy of our program by comparing our results to the given information in the initial dataset. In total, we used 180 tracks, and ~17,000 individual notes for our project. This might not seem like a lot of data initially, but it is important to note that the sequences of notes were our focus, rather than individual notes.

We intended to use a pre-existing algorithm for monophonic pitch detection so that we could focus most of our efforts on converting the signals into readable, accurate guitar tabs. Work by Makhmutov *et al.* [16] employ the Fast Fourier Transform (FFT) to detect pitches in monophonic vocal signals alongside various Monte Carlo methods for tempo detection [2]. While we were initially planning on implementing one of these algorithms, we describe a shift in our project that diverted us away from monophonic pitch detection (Section 2.2).

Finally, the results of our project are fairly interesting. The accuracies of each model described in this section are summarized in Figure 2 below. Accuracy in our case refers to the number of correct string predictions.

| Model | Accuracy (%) | Average Distance Cost (Manhattan) |
|---|---|---|
| **Baseline** | 30 | 137 |
| **Naïve Shortest Distance** | 40 | 110 |
| **Improved Shortest Distance** | 47 | 96 |
| **MLP Classifier** | 64 | 143 |

**Figure 2:** breakdown of each algorithmic approach by percentage accuracy and average distance cost.

Overall, the accuracies are well aligned with our initial predictions; the baseline model would obviously be the worst, and we were pleased to see that as each model increased in complexity, it also increased in accuracy. It was surprising to see that this was not the case with the average cost associated with each function; the improved shortest distance approach was the best approach in terms of cost. The average distance cost of our target data was ~126, from the results table above we can see that only the Naïve Shortest Distance, and Improved Shortest Distance approach reduces that figure.

This suggests that our distance metric is flawed; as accuracy improves, the playability of the tabs should also improve, otherwise we have not optimized the performance of the overall final product.Wee essentially wanted to not penalize for frets that are next to each other on the same string, and not penalize from going from an open string to any fret. Additionally, taking into account hand position would have been ideal. However, given that we did not spend much time researching the best way to implement cost in terms of

playing guitar notes, we are fairly pleased with these results. Although they are not ideal, we can clearly see which areas need improvement.

### 2.2 Major Challenges

The first major challenge encountered was related to our dataset. Some tracks contained polyphonic signals, which was well outside the scope of this project. We circumnavigated this problem by removing the tracks that had too many polyphonic signals. If a track had more than 20% of polyphonic signals then we excluded it from our datasets. This was a primitive approach, but as we did not anticipate this issue, we had to come up with a solution quickly to continue with the project. A more advanced solution might have been removing the notes that are polyphonic to convert it into a monophonic input, but we weren't sure how to do this correctly. Risking doing this incorrectly could have introduced more problems rather than solving the initial issue.

As mentioned, we also realized along the way that accuracy is a subjective measure. We would need an experienced guitarist to come in and confirm whether or not our tabs are indeed accurate. This caused us to consider re-evaluating our project, however by this point it was too late in the process. This also means that future iterations of this project will require significant improvements to the measurement of accuracy.

Finally, we initially intended to construct a full pipeline, that would start with monophonic pitch detection, and end with an automatically generated guitar tablature. However, we realized that there was already an incredible amount of work and research that had been done in this area. This part of the initially intended pipeline was also fairly intensive, and we realized that we might not get very far if we focused on this part of the pipeline; part of the problem is that this was a linear process. Instead, we shifted the focus of our project over to different phases of the pipeline that interested us more, which was the output of the pitch and beat detection to the generation of the tab. We also focused on the comparison between the performance of various algorithms, summarized in Figure 2, and dedicated most of our efforts into implementing them.

With all these challenges, we were able to demonstrate successful tab generation. Figure 3 is an example of one of our final tabs created; as you can see, it is more reasonable than one of our earlier tabs shown in Figure 1.

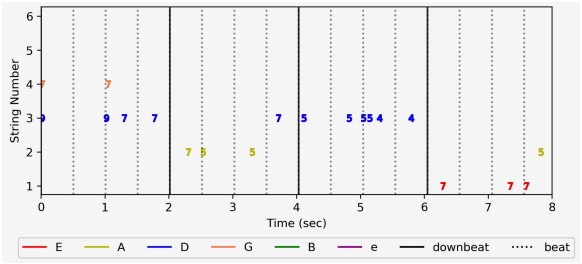


**Figure 3:** Sample final tablature generated

### 2.3 Related Works

Previous studies such as one by Choi *et al.* employ the use of deep learning with Keras for audio pre-processing [14]. Although this work is not identical to our project idea there are some similarities in early sections of the study that involved the processing of monophonic signals. This study helped us come up with the ideas of using machine learning and comparing ML models to other models.

While reviewing the related literature, we also found a study by Hori et al. which implements automatic tab generation by using Hidden Markov Models [10]. The finger positions on the fretboard are represented by the hidden states of the model. The authors note that even if the precise fingering cannot be determined by the melody (as there are multiple possible ways to play a note), a "most probable fingering can still be estimated" using maximum likelihood estimation. This is a model we may explore as well, as it addresses our first challenge described in Section 1.2 of there being multiple ways to play the same pitch on a guitar. Another way to approach this problem is by using A-star search, where vertices in a graph could represent potential combinations of frets and strings, and the A-star algorithm finds the best path (in terms of hand position) through the melody [12]. If we were to use this approach we may have to add a constraint on melody length. This is because the state space (all possible ways to play the melody on a guitar) would be quite large given the redundancy of guitar notes. Having longer melodies would then require a lot of space. While we were very intrigued by these methods, we did not reach the point of attempting to implement a Hidden Markov Model. This is something we would like to do as part of future work towards this project,

as we see a potential for significant improvement and incorporation into the suggestions from experienced guitarists.

In the same thread, we considered how other graph searching algorithms would impact our results. While A-star search would likely generate a fairly accurate tab (in terms of playability and best fingering positions), it takes up a lot of space unless we add a constraint on melody length. Another option that was considered was some variation of depth first search (DFS). In the generated search tree, the complete tab for the melody will always be at the bottom of the tree, however, this does not guarantee the *best* tab possible, it just guarantees a *complete* tab. Again, we did not reach the point of exploring this option, but it would be interesting to implement and add to our current comparisons.

Another interesting approach that we found was using deep learning for automatic music generation discussed by Garcia *et al*.[13]. Although we did not take this approach, we thought it was really interesting, and were taking some considerations from the paper, such as concerns regarding overfitting and accuracy. The authors present a model that generates musical notation (in standard notation as opposed to guitar tabs) automatically. The paper references other relevant studies that treated music notation akin to natural language. Although this approach would be very interesting, our initial thoughts were that training a model to recognize all the symbols and behaviours of music notation might make the model prone to overfitting. After reading the study in its entirety, we realized the authors also addressed this as an issue. However, for the scope of a small model that generates simple guitar tabs, overfitting may not necessarily be the primary concern. Given the constraints of our project, we were more concerned with accuracy rather than the model's ability to generalize to more complex inputs. In fact, Benetos *et al.* study the current limitations of automatic music transcription and conclude that "one way to overcome the limited performance" would be to "tailor algorithms to specific use cases" [8].

While we learnt an incredible amount by reading this study, we also realize that we are not at the stage where we should be deeply concerned with accuracy or concepts like overfitting. Our final product still has other major flaws such as a cost function that does not appear to accurately represent the cost of hand movements. Once we are able to rectify these issues we can move on to accuracy concerns. As described earlier, we ended up having the opposite problem, where our model was underfitting, which again suggests that our concerns regarding overfitted models was quite premature.

Interestingly, deep learning has also been used to generate melodies as well as the corresponding guitar tabs. Models were trained based on the ability to generate a coherent melody (rather than a disjoint series of notes), as well as the ability to generate "rhythmic groove" [19]. While generating melodies and rhythms are outside the scope of our project, learning about the tab-generation portion of this study helped us arrive at our final project idea by putting our project's scope into perspective.

One of our major future goals for this project will be to complete the pipeline. We still have hopes that we will be able to build a complete platform that employs monophonic pitch detection to generate a tab directly from an audio signal. We will also want to implement beat detection, but with help from the user. For example, we could ask the user to set the tempo beforehand, and then we would use that information to quantize note durations to the beat. In order to do all of this, we would also want to implement a proper user interface for the system.

To accomplish this, we would feel fairly confident in using the autocorrelation function to find individual pitches from our input. Related work that also used autocorrelation to detect pitches then used a peak search method to recognize notes [17]. Overall, the peak search method in the study seemed to make sense given the content covered in CSC 475 thus far. A similar study by Monti *et al.* reveals that autocorrelation worked very well for monophonic music transcription also, stating the method was "simple and reliable", which is very beneficial for our project [11]. This study also discusses a publicly available tool that generates sheet music, however this tool is no longer accessible. We were hoping to take a look at the implementation of this tool when we had plans to construct the full pipeline.

In the future, we will try pitch detection using both FFT and autocorrelation to see which one performs better in our overall model. While this might be a bit difficult to do, one of our initial goals was to generate a model with a very good accuracy. Trying different pitch detection algorithms would also help us identify where our model is weak. For example, if changing the pitch detection algorithm has minimal impact on the overall accuracy of the tabs, then we know that the flaws lie elsewhere.

Lastly, we found one approach to guitar tab generation that was too complex to implement for the scope of

this project, but was very interesting. Tuohy *et al.* use a distributed genetic algorithm to create the tabs themselves, and then use an artificial neural network to help assign finger positions to each note [6]. After some further research we discovered a genetic algorithm is one that covers a very large, but optimizable search space such as some type of scheduling problem. A *distributed* genetic algorithm is one that runs on multiple machines at the same time to generate a solution. While this is not relevant to our current project it was an interesting side note, and reading the study allowed us to see how both simpler and very complex algorithms are applicable to the same area in MIR.

### 2.4 Roles and Reflections

Both group members contributed equally to all the components of this project (except for one member unable to attend the presentation due to an exam conflict). While there were some difficulties faced we felt that we were able to overcome them and settle on a final project that we were both satisfied with, and interested in.

Initially as a minimum, we wanted to be able to generate tabs using any approach (even with just our baseline model) and compare the result to a machine learning approach with a basic percentage accuracy. In this scenario we didn't place any constraints on our accuracy, we simply aimed to have a working model, regardless of its performance.

For our target project, we wanted to do the same as before, but with a more intelligent approach that results in an accuracy > 50%. Instead of just taking the lowest fret possible for the desired note, our model should have been able to decide which string and fret were most probable based on the cost of moving to the next string and fret. This implied that we should have developed a better cost function using these heuristics, rather than just a percent correct approach. At the time, we felt like this was not too unrealistic. We found a very interesting project by Hsu *et al.* that used a deep learning approach for sheet music generation and their model had accuracies of almost 80% [20]. Given our lack of expertise and experience, we are hoping that 50% is a reasonable goal.

Finally, in an ideal world, we would have liked to try this using an HMM where the hidden states represent combinations of fret-string combinations [10]. We would have produced a project with an accuracy > 70% and had the chance to have a few guitarists, perhaps just our friends, try it out and see if the tablature generated is playable. In this scenario, the model produced would have been relevant to the motivations described in Section 1; that is, it should have benefited real guitar players and have some impact outside the scope of a course-project.

While we did not get to our very ambitious goals of implementing an HMM, and having guitarists test it out, we did achieve creating more intelligent approaches than our baseline model. We also successfully completed a machine learning approach and we were able to decipher a lot by comparing the performances of these algorithms. Most excitingly, we were able to produce somewhat playable tabs. Even though there was not a huge improvement in accuracy between our baseline model and naïve shortest path model, the improved shortest path approach generated somewhat playable tabs already.

We learnt a lot from this project. Aside from exposure to new algorithms, libraries, and features, we benefitted a lot as there was much critical thinking involved. For example, by looking at our results, we were able to decipher that something was wrong with our cost function. Knowing this, we considered several factors that our project plan did not account for, such as natural resting hand positions, playing styles, and perhaps even how playing styles will differ based on the genre of the track. We also were exposed to skills that will be beneficial in the future, such as quick problem-solving. When we ran into issues with our dataset, we picked a (rather primitive) work around which seemed like the best option at the time. Although in hindsight, there may have been a better option, making the choice to eliminate tracks with too many polyphonic signals ultimately allowed us to create a final product that was functional.


## ACKNOWLEDGEMENTS

Both members of our group would like to thank Dr. George Tzanetakis for a very interesting semester of CSC 475 and for the chance to work on such an interesting project.

## REFERENCES


[1] Andrew Wiggins and Youngmoo Kim. 2019. GUITAR TABLATURE ESTIMATION WITH A CONVOLUTIONAL NEURAL NETWORK. *ISMIR Conference* (November 2019), 284-286).

[2] A.T. Cemgil and B. Kappen. 2003. Monte Carlo methods for Tempo Tracking and rhythm quantization. *Journal of Artificial Intelligence Research* 18 (2003), 45–81. DOI:http://dx.doi.org/10.1613/jair.1121

[3] Brata and I.D. Darmawan. 2021. Comparative study of pitch detection algorithm to detect traditional Balinese music tones with various raw materials. *Journal of Physics: Conference Series* 1722, 1 (2021), 012071. DOI:http://dx.doi.org/10.1088/1742-6596/1722/1/012071

[4] Carlos de Obaldia and Udo Zolzer. IMPROVING MONOPHONIC PITCH DETECTION USING THE ACF AND SIMPLE HEURISTICS. *International Conference on Digital Audio Effects*, 2–6.

[5] David Baker. 2014. Visually impaired musicians' insights: Narratives of childhood, lifelong learning and musical participation. *British Journal of Music Education* 31, 2 (2014), 113–135. DOI:http://dx.doi.org/10.1017/s0265051714000072

[6] Daniel R. Tuohy and W.D. Potter. 2006. Generating guitar tablature with LHF notation via DGA and ann. *Advances in Applied Artificial Intelligence* (2006), 244–253. DOI:http://dx.doi.org/10.1007/11779568_28

[7] Elias Mistler. 2017. Generating Guitar Tablatures with Neural Networks. *School of Informatics, University of Edinburgh* (2017), 4–15.

[8] Emmanouil Benetos, Simon Dixon, Dimitrios Giannoulis, Holger Kirchhoff, and Anssi Klapuri. 2013. Automatic Music Transcription: Challenges and Future Directions. *Journal of Intelligent Information Systems* 41, 3 (2013), 407–434. DOI:http://dx.doi.org/10.1007/s10844-013-0258-3

[9] Eric Larson and Ross Maddox. Real-Time Time-Domain Pitch Tracking Using Wavelets.

[10] Gen Hori, Hirokazu Kameoka, and Shigeki Sagayama. 2013. Input-output HMM applied to automatic arrangement for guitars. *Journal of Information Processing* 21, 2 (2013), 264–271. DOI:http://dx.doi.org/10.2197/ipsjjip.21.264

[11] Giuliano Monti and Mark Sandler. 2000. MONOPHONIC TRANSCRIPTION WITH AUTOCORRELATION. *Proceedings of the COST G-6 Conference on Digital Audio Effects* (December 2000). DOI:http://dx.doi.org/https://www.dafx.de/paper-archive/2000/pdf/Monti_DAFX00poster.pdf

[12] Gregory Burlet and Ichiro Fujinaga. 2013. ROBOTABA GUITAR TABLATURE TRANSCRIPTION FRAMEWORK. *The International Society for Music Information Retrieval (ISMIR)* (2013). DOI:http://dx.doi.org/https://ismir2013.ismir.net/wp-content/uploads/2013/09/217_Paper.pdf

[13] Juan Carlos García and Emilio Serrano. 2018. Automatic Music Generation by deep learning. *Distributed Computing and Artificial Intelligence, 15th International Conference* (2018), 284–291. DOI:http://dx.doi.org/10.1007/978-3-319-94649-8_34

[14] Keunwoo Choi, Deokjin Joo, and Juho Kim. Kapre: On-GPU Audio Preprocessing Layers for a Quick Implementation of Deep Neural Network Models with Keras. DOI:http://dx.doi.org/https://arxiv.org/pdf/1706.05781.pdf

[15] Lance Alcabasa and Nelson Marcos. 2012. Automatic Guitar Music transcription. *2012 International Conference on Advanced Computer Science Applications and Technologies (ACSAT)* (2012). DOI:http://dx.doi.org/10.1109/acsat.2012.78

[16] Munir Makhmutov, Joseph Alexander Brown, Manuel Mazzara, and Leonard Johard. 2017. Momos-MT: Mobile monophonic system for music transcription. *Proceedings of the Symposium on Applied Computing* (2017). DOI:http://dx.doi.org/10.1145/3019612.3019723

[17] Mursel Onder, Aydin Akan, and Semih Bingol. PITCH DETECTION FOR MONOPHONIC MUSICAL NOTES. *Istanbul University, Department of Electrical and Eelectronics Engineering*. DOI:http://dx.doi.org/http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.583.7390&rep=rep1&type=pdf

[18] Qingyang Xi, Rachel M. Bittner, Johan Pauwels, Xuzhou Ye, and Juan P. Bello. 2019. Guitarset. (August 2019). Retrieved February 1, 2022 from https://zenodo.org/record/3371780

[19] Yu-Hua Chen, Yu-Hsiang Huang, Wen-Yi Hsiao, and Yi-Hsuan Yang. Automatic Composition of Guitar Tabs by Transformers and Groove Modeling. DOI:http://dx.doi.org/https://arxiv.org/abs/2008.01431

[20] Yu-Lun Hsu, Chi-Po Lin, Bo-Chen Lin, Hsu-Chan Kuo, Wen-Huang Cheng, and Min-Chun Hu. 2017. DeepSheet: A sheet music generator based on Deep Learning. *2017 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)* (2017). DOI:http://dx.doi.org/10.1109/icmew.2017.8026272