

GUITAR NOTE DETECTION USING A CONVOLUTIONAL NEURAL NETWORK MODEL

Ilija Jovanović¹ Dušan Terzić² Perica Štrbac³

Abstract: The paper presents an application for guitar note detection using a convolutional neural network (CNN). The choice to use a CNN was made based on the existing usage and application of such deep-learning models in speech recognition. The initial step in the processing of the audio recording is filtering out irrelevant low and high frequencies. That is followed by the detection of segments and recognizing which note is present in each of those segments. The segments are converted into mel spectrograms, and used as input for the CNN model. Based on CNN's predictions, an array of recognized notes is generated according to the musical letter notation system. The presented application is specifically designed for processing audio recordings of electric or acoustic guitars. The CNN model was successfully incorporated into an application written in the Python programming language. Executing this application produced experimental results, which are shown and described in the paper. These results confirm the accuracy and reliability of the note detection model provided by the convolutional neural network.

Key words: guitar note detection, deep learning, convolutional neural network, mel-spectrogram, modeling, analysis

1. INTRODUCTION

Following multiple iterations of training, this model was transformed into a real system as a web application written in Python. Python was chosen because it provides a wide range of modules and libraries for working with audio files, neural networks, graphics, and the Flask web framework [1], [2]. Additionally, Python was selected as program execution speed was not a critical factor for this project, making its ease of use and versatility ideal for implementation [3].

The modules and libraries used include soundfile, cv2, numpy, matplotlib (pyplot), and tf.keras. The module tf.keras from the Tensorflow library allows efficient implementation and interaction with artificial neural networks in Python programs. The sound recordings that were chosen as inputs were processed by a program that detected the notes played as segments of the file. The output of the program is an array of string values that represent the segments as notes using letter notation.

Once the model was integrated into a Flask-based web application, the end-user could send an HTTP request to the Flask server. The request contains three parts, an audio file in the .wav format, a decibel level at which the recording should be analyzed, and a value that indicates which process should be used in the analysis. The file is then processed and the output array of note names, as well as a loudness chart image, are returned as a response to the initial request. The array is then visually represented in the form of guitar tablatures. The loudness chart can be viewed as an image and used to determine the proper decibel level.

2. THE CNN NOTE RECOGNITION MODEL

This chapter briefly covers important digital audio signal topics and describes a CNN model that recognizes notes in an audio recording based on the recording's mel spectrogram and labels them according to the music letter notation system [4],[5].

¹ Ilija Jovanović, Academy of Technical and Art Applied Studies Belgrade, 24 Starine Novaka St., email: ilijaj@viser.edu.rs

² Dušan Terzić, Academy of Technical and Art Applied Studies Belgrade, 24 Starine Novaka St., email: terzicd@viser.edu.rs

³ Perica Štrbac, Academy of Technical and Art Applied Studies Belgrade, 24 Starine Novaka St., email: pericas@viser.edu.rs

2.1. Brief digital audio signals overview

Due to the difference in how digital audio and analog audio levels are measured, the unit dBFS (decibels relative to full scale) is used for digital audio signals. The loudest level when using dBFS is zero. Therefore, values lower than zero are passed to the application as a parameter before processing.

A mel spectrogram is a spectrogram variation that shows the frequencies present in an audio signal at different points in time, along with their loudness level (dB level is represented by color value). The frequency values (in Hz, on the Y axis) are on a logarithmic scale. The primary fields mel spectrograms are used in are machine learning models and speech processing.

2.2. CNN note recognition model

The first layer is an Input Layer, which has no weights and biases and has dimensions of $128 \times 128 \times 1$. The second layer is a Convolutional Layer with 32 filters, a kernel size of 3×3 , and a ReLU activation function [6]. The third layer is a Max Pooling Layer with a window size of 2×2 , effectively reducing the feature map width and height by half while keeping the most important features [7]. The fourth layer is another Convolutional Layer, this time with 64 filters, a kernel size of 3×3 , and a ReLU activation function. The fifth layer is identical to the third layer, being a Max Pooling Layer with a window size of 2×2 . The sixth layer is another Convolutional Layer with the same kernel size and activation function but with 128 filters. The seventh layer is a Flattening Layer, which converts the 2D feature map into a 1D vector. The eighth layer is a Dense (Fully Connected) Layer with 64 neurons and a ReLU activation function. The ninth and final layer is another Dense Layer with 30 neurons representing 30 possible notes and a Softmax activation function, converting the values from each neuron into the probability of the input belonging to a specific note.

The model was trained on a dataset containing over 1,500 audio files of individual notes, specifically created for this project. The dataset was split into training and test datasets with an 80:20 ratio. The model was initially trained for 20 epochs with a batch size of 32. However, it was determined that the best results were achieved after 3 epochs, and further training could lead to overfitting (Figure 1). Adaptive Moment Estimation (Adam) optimizer was used to adjust the model's weights during training. Since the true labels are represented as integers, the sparse categorical cross-entropy loss function was used.

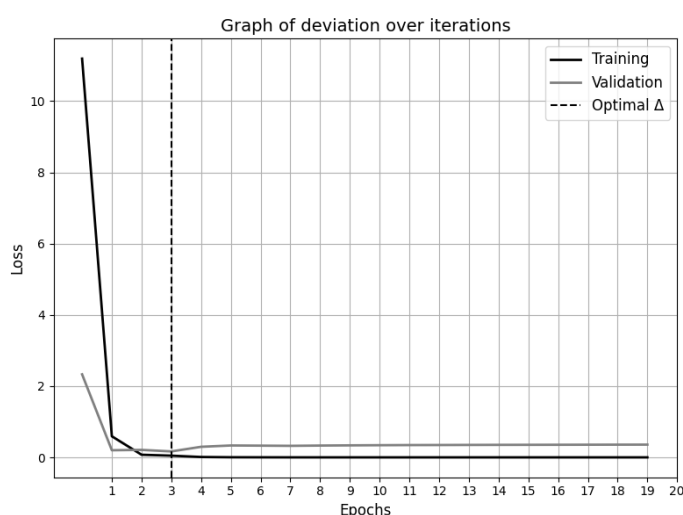


Figure 1 – Filtered Mel Spectrogram of input audio recording

3. INPUTS AND OUTPUTS OF THE PROGRAM

In standard guitar tuning the strings are tuned in the following order from lowest to highest: E2, A2, D3, G3, H3, E4. Since the frequency of the lowest note on the guitar in standard tuning is 82 Hz

and the third octave of the highest detectable note (A4) is at 3520 Hz (A7) the input is filtered below 80Hz and above 3600 Hz to reduce input size and make training more efficient.

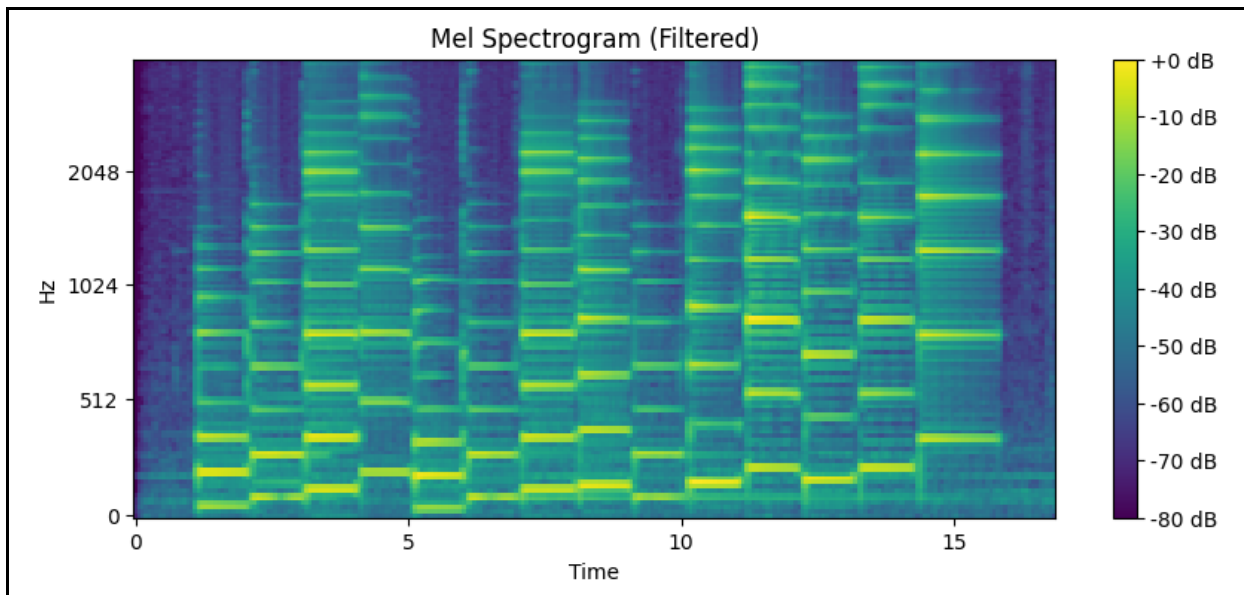


Figure 2 – Filtered Mel Spectrogram of input audio recording

Recordings of electrical and acoustic guitar melodies were used as inputs during development. All of the recordings are in the WAV format. The WAV (Waveform Audio File Format) format provides a way to store digital audio data without compression, thus eliminating any loss of quality or any other changes that may occur when other formats are used for audio recordings.

The output is an array containing the letter names of the detected notes. Based on the CNN model's prediction of which note the mel-spectrogram is representing, an integer value is returned and used to find the matching note letter. The note letter will then be added to the output array. Note letters and their corresponding numbers are given in Table 1.

Table 1 – Note letter names

Note letter names and numeric value pairs
(E2, 0) (F2, 1) (F#2, 2) (G2, 3) (G#2, 4) (A2, 5) (A#2, 6) (H2, 7) (C3, 8) (C#3, 9) (D3, 10) (D#3, 11) (E3, 12) (F3, 13) (F#3, 14) (G3, 15) (G#3, 16) (A3, 17) (A#3, 18) (H3, 19) (C4, 20) (C#4, 21) (D4, 22) (D#4, 23) (E4, 24) (F4, 25) (F#4, 26) (G4, 27) (G#4, 28) (A4, 29)

4. EXPERIMENTAL RESULTS

To verify the output of this model, it was integrated into a Flask web application which visually represents the output results in the form of a loudness chart that shows an image based on the first part of the output (a byte buffer containing image data in PNG format) and a guitar tablature based on the array of note names (Figure 3).

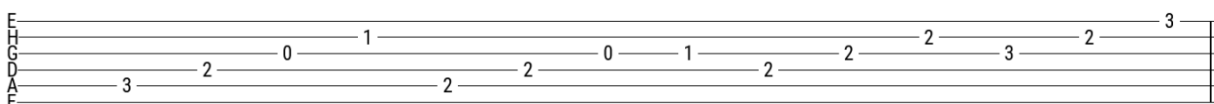


Figure 3 – The tablature representation of an input recording

The first input parameter is an audio recording of a guitar melody in the WAV format file. The second input parameter is the loudness threshold (measured in dBFS) which has a numeric value. The third and final input parameter is a value that indicates whether only the amplitude or both the amplitude and note onset should be used as parameters during the note detection. Using only the amplitude has proven to be more accurate for melodies with a slower tempo while using both the amplitude and note onset detection has proven to be more accurate for melodies with a faster tempo.

The audio signal of the input recording is represented using the matplotlib library (Figure 4). The changes in loudness over time are depicted, and each detected note segment is visually marked at the beginning and end points in time between which the segment occurs in the recording. The dBFS threshold at which the recorded input is analyzed is also represented on the loudness chart. The figure data from the matplotlib chart is then written into a byte buffer as an image in the PNG format. The data from that byte buffer is the first of two parts of the output. The second part is the note letter array.

In the experimental example which is presented in Figures 2, 3, and 4, an audio recording, which contains 14 played notes, was used along with the loudness threshold value of -20 dBFS. The note detected 14 segments and the loudness threshold can be seen on the loudness chart (Figure 3). The following notes were played in the recording: C3, E3, G3, C4, H2, E3, G3, G#3, E3, A3, C#4, A#3, C4, G4.

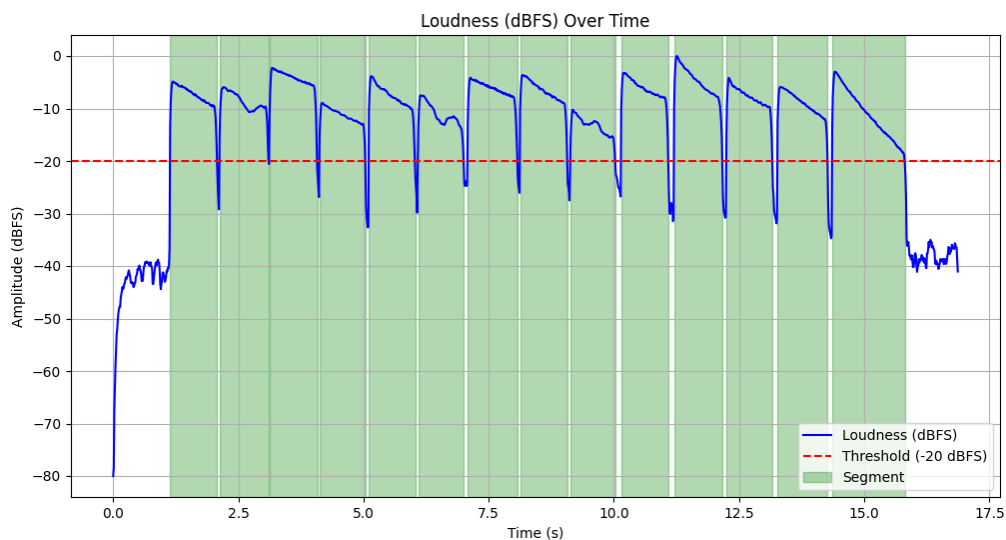


Figure 4 – The visual depiction of the analyzed audio recording

5. CONCLUSION

In this paper, a machine learning approach to musical note detection is proposed, utilizing a convolutional neural network (CNN) as an apparatus for modeling, analysis, and prediction. The proposed CNN model processes mel spectrograms, which represent the digital transformation of audio signals into the frequency domain, as its input. The application performs a series of operations, including filtering high and low frequencies, splitting audio files into individual note segments, converting segments into mel spectrograms, feeding the spectrograms into the CNN model for classification, and generating an array of detected musical notes according to the letter notation system. The CNN model was integrated into a Python-based web application, using a lightweight web framework provided by the Flask module. Through the execution of this application, the correctness and reliability of the model were verified using experimental data. Results obtained from testing demonstrate a high accuracy rate in note detection, confirming the model's effectiveness for analyzing audio recordings of electric and acoustic guitars. Additionally, the application allows for the

processing and display of detected notes in real-time, further validating the practical applicability of the model. The paper also includes an example of spectrogram visualizations, detected segments, and detected notes presented as guitar tablatures. These results highlight the robustness of the proposed approach and its potential for broader applications. Future research directions will explore the expansion of the CNN model's capabilities, focusing on generalizing its scope and investigating its adaptability to more complex scenarios and diverse datasets.

6. REFERENCES

- [1] Vyshnavi V. R.; Malik A.: *Efficient Way of Web Development Using Python and Flask*, International Journal of Recent Research Aspects Vol. 6, Issue 2, pp. 16-19, June 2019
- [2] Sayeth Saabith A.L.; Vinothraj T.; Fareez M.M.M.: *Popular Python Libraries And Their Application Domains*, International Journal of Advance Engineering and Research Development Vol. 7, Issue 11, pp. 21-25, November 2020
- [3] J. Tait, J., Ripke, T., Roger, L., Matsuo, T.: *Comparing Python and C++ Efficiency Through Sorting*, 2018 International Conference on Computational Science and Computational Intelligence (CSCI), Las Vegas, NV, USA, 2018, pp. 864-871.
- [4] Egmont-Petersen, M.; de Ridder, D.; & Handels, H. *Image processing with neural networks—a review*, Pattern Recognition, 35(10): 2279–2301, August 2001, doi:10.1016/s0031-3203(01)00178-9
- [5] Li, Z., Liu, F., Yang, W., Peng, S., & Zhou, J: *A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects*. IEEE Transactions on Neural Networks and Learning Systems, pp. 1–21, July 2021, doi:10.1109/tnnls.2021.3084827
- [6] Rasamoelina, A. D.; Adjailia, F.; Sincak, P.: *A Review of Activation Function for Artificial Neural Network*. IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), pp. 1-5, June 2020, doi:10.1109/sami48414.2020.9108717
- [7] Gholamalinezhad, H.; Khosravi, H.: *Pooling Methods in Deep Neural Networks, a Review*, pp. 2-5, September 2020