Hindawi
Mobile Information Systems
Volume 2022, Article ID 8544765, 11 pages
https://doi.org/10.1155/2022/8544765

Research Article

Transfer Learning for Audio Waveform to Guitar Chord Spectrograms Using the Convolution Neural Network

Yogesh Jadhav,[1] **Ashish Patel**,[2] **Rutvij H. Jhaveri**,[3] **and Roshani Raut**[4]

[1]*School of Technology Management & Engineering, SVKM's NMIMS University, Navi Mumbai, India*
[2]*Department of Computer Engineering, Shri Sad Vidya Mandal Institute of Technology (SVMIT), Bharuch, India*
[3]*Department of Computer Science and Engineering, School of Technology, Pandit Deendayal Energy University, Raysan, India*
[4]*Pimpri Chinchwad College of Engineering, Savitribai Phule Pune University, Pune, India*

Correspondence should be addressed to Rutvij H. Jhaveri; rutvij.jhaveri@sot.pdpu.ac.in

Received 26 April 2022; Revised 23 June 2022; Accepted 23 July 2022; Published 31 August 2022

Academic Editor: Saqib Hakak

Automatic chord recognition has always been approached as a broad music audition task. The desired output is a succession of time-aligned discrete chord symbols, such as GMaj and Asus2. Automatic music transcription is the process of converting a musical recording into a human-readable and interpretable representation. When dealing with polyphonic sounds or removing certain limits, automatic music transcription remains a difficult undertaking. A guitar, for example, presents a greater challenge, as guitarists can play the same note in a variety of places. The study makes use of CNN functionality to generate the guitar tab; initially, the constant-Q transform was used to turn the input audio file into short time spectrograms that the CNN model utilises to analyse the chord. The paper developed a method for extracting chord sequences and notes from audio recordings of solo guitar performances. For intervals in the supplied audio, the proposed approach outputs chord names and fret-board notes. The model described here has been refined to achieve an accuracy of 88.7%. The model's ability to properly tag audio clips is an incredible advancement.

1. Introduction

The guitar is one of the world's most popular instruments. Professional musicians play the guitar, but it is also an excellent instrument for anybody who enjoys music and wants to play it freely, even if they have no prior understanding of music theory. Although guitars come in a variety of styles, the bulk of them have six strings. Modern guitar strings are 65 centimetres in length and tuned to the following pitches: $E2 = 82$ Hz, $A2 = 110$ Hz, $D3 = 147$ Hz, $G3 = 196$ Hz, and $E4 = 330$ Hz. The strings of a guitar might be of nylon or steel, depending on the model. In this work, we shall use the phrases classical guitar and acoustic guitar [1, 2] to refer to nylon-stringed guitars that are not electric guitars.

Each guitar string is slightly unique, not just in diameter but also in how the strings are produced. On a guitar, the same pitch can be performed using a variety of string, fret, and finger combinations. This paper will cover the string/fret combinations that may be utilised to play the various notes on classical, acoustic, and electric guitars with six strings. However, the technique and concepts demonstrated here may be applied to any other guitar with any number of strings.

Composing music for the guitar can be done in two ways: using a classical score or using guitar tablature. Different amounts of time, fretting, and fingering information are provided by these two alternatives. For performing a certain score, there is no one-of-a-kind tablature. A guitar string is represented by each line in the tablature. The location of the number indicates the string to pluck, and the number itself denotes the fret number in the guitar neck where the associated string must be pushed with a finger of the left hand to create the required note when plucking the string with the right hand. It is worth noticing that when the vibrating component of the string gets shorter, the number increases.

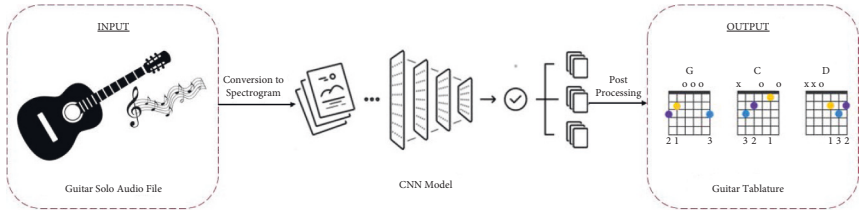

Figure 1: Visualization of the guitar tablature pipeline.

The string should not be pushed at any fret on the guitar neck; hence, the tablature numbering starts with 0. As can be seen, the identical notes may be played on different strings of the guitar. Even though the notes performed are same, the sound will vary somewhat, changing the difficulty of the performance. Top guitarists have more options for string and fret combinations than normal guitarists, and their selections establish a particular sonority for the guitar compositions they perform, as well as a comfortable fingering technique. [3].

Many guitarists are familiar with tablature, a musical score style for string instruments such as the guitar and bass. Tablature employs numbers to represent the strings and fret locations, allowing players to play the guitar naturally even if they do not know much about music. Because a guitar may generate the same pitch with different strings, many fingerings for a series of pitches are feasible. Tablature removes the ambiguity of fingerings, making it easier for players to play the guitar. As a result, many guitar players, particularly novices, find tablature useful when playing the instrument. [4, 5].

Despite the fact that tablature can motivate most guitarists to practise, it is not always available. The suggested technology aids in the creation of tablature from the selected audio. Figure 1 represents visualization of the pipeline.The authors described a method for extracting chord sequences and notes from audio recordings of solo guitar performances. The proposed approach outputs chord names and fret-board notes for intervals in the supplied audio and achieves the improved result. The progress can be credited to the creation of musical models expressly for the guitar and the training of acoustic models employing guitar acoustic material. This demonstrates how optimizing music transcription algorithms for more specific circumstances can result in significant performance gains. Due to the correctness of the suggested system, it is a viable contender for usage by the target audience.

The rest of the article is structured as follows: Section 2 discusses the current state of automated guitar tablature transcription. Section 3 discusses how the automated chords were generated using a convolutional neural network. Section 4 examines the study's findings and implications, followed by Section 5's conclusion.

2. Related Work

For creating spectrograms to be further processed by the CNN model, the constant-Q transform (CQT) is the chosen transform technique. CQT is a time-frequency representation in which the frequency bins are geometrically separated and all bins have equal Q-factors (ratios of centre frequencies to band-widths) [1, 6]. Since the auditory signal to be processed is musical in nature, we are preferring the CQT over STFT (short time Fourier transform) to understand this shift of preference from the more traditional STFT to CQT; we must understand at [7] how musical notes as defined in Table 1.

In the NOTE column, the letter on the left denotes the musical note, while the number on the right reflects the current octave. A graph of the frequencies (Hz) for the first six octaves of the musical note C can be found in Figure 2.

The frequency of each subsequent octave is double that of the preceding octave. We know that the frequency must double every twelve notes since an octave spans twelve notes, and it can be expressed by the following formula:

$$F_k = 440Hz \times 2^{k/12}. \quad (1)$$

Because of its exponential structure, the constant-Q transform is better suited for fitting musical data than the Fourier transform since its output is amplitude versus log frequency. Furthermore, the constant-Q transform's accuracy is similar to the logarithmic scale and mimics the human ear's frequency resolution, with higher frequency resolution at lower frequencies and lower frequency resolution at higher frequencies.

The mentioned paper justifies the use of CQT over the traditional STFT. We will be using the librosa Python library in this project to generate CQT of the guitar solo audio, which will take care of all the computational complexities efficiently. However, there is a plethora of parameters that must be considered when using this transform in order to produce a spectrogram that makes sense, all of which must be set with the nature of the signal (musical in this case) to be processed in mind. The parameters to be used in our case as shown in Table 2 are as follows:

2.1. GUITARSET: A Dataset for Guitar Transcription. To tackle the challenge of guitar transcription, we will require a collection of guitar recordings containing sets of annotated audio data. The GuitarSet, a dataset that comprises high-quality guitar recordings as well as significant annotations and metadata, is presented in this study. By recording guitars with a hexaphonic pickup, it provides separate recordings from each of the six strings. The dataset contains recordings of acoustic guitar excerpts as well as time-aligned annotations on string and fret positions, chords, beats, downbeats, and playing style, all of which are necessary to train our

Table 1: Musical notes and octave values.

	Oct0	Oct1	Oct2	Oct3	Oct4	Oct5	Oct6	Oct7	Oct8	Oct9	Oct10
C	16.35	32.70	65.41	130.81	261.63	523.25	1046.50	2093.00	4186.01	8372.02	16744.04
C#	17.32	34.65	69.30	138.59	277.18	554.37	1108.73	2217.46	4434.92	8869.84	17739.69
D	18.35	36.71	73.42	146.83	293.66	587.33	1174.66	2349.32	4698.64	9397.27	18794.55
D#	19.45	38.89	77.78	155.56	311.13	622.25	1244.51	2489.02	4978.03	9956.06	19912.13
E	20.60	41.20	82.41	164.81	329.63	659.26	1318.51	2637.02	5274.04	10548.08	
F	21.83	43.65	87.31	174.61	349.23	698.46	1396.91	2793.83	5587.65	11175.30	
F#	23.12	46.25	92.50	185.00	369.99	739.99	1479.98	2959.96	5919.91	11839.82	
G	24.50	49.00	98.00	196.00	392.00	783.99	1567.98	3135.96	6271.93	12543.86	
G#	25.96	51.91	103.83	207.65	415.30	830.61	1661.22	3322.44	6644.88	13289.75	
A	27.50	55.00	110.00	220.00	440.00	880.00	1760.00	3520.00	7040.00	14080.00	
A#	29.14	58.27	116.54	233.08	466.16	932.33	1864.66	3729.31	7458.62	14917.24	
B	30.87	61.74	123.47	246.94	493.88	987.77	1975.53	3951.07	7902.13	15804.26	

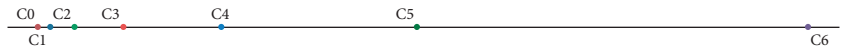

Figure 2: First six octaves of the musical note C.

Table 2: List of the parameters with values.

Parameters	Value
Sampling rate	44100
Number of samples between successive CQT columns	1024
Number of frequency bins	96
Number of bins per octave	12

Table 3: Chord classes.

Ten classes of chords
A major
A minor
B minor
C major
D major
D minor
E major
E minor
F major
G major

model [8–10]. JAMS file format is used to store the rich collection of annotations for this dataset. It is a JSON annotated music specification. It provides JSON structure for generic annotations. It stores multiple annotations per file. For each recording, the JAMS file contains annotations for tempo, key, and style (metadata), beats and downbeats (inferred from the click track), instructed chords (from the lead-sheets), performed chords (via automatic estimation), note-level transcription, including string and fret position (via automatic estimation), onsets (via annotation), offsets (via automatic estimation), and pitch contour for each note (via validated automatic estimation) [11–14]. The annotations of our interest are mentioned as follows:

(i) Tempo

(ii) Note-level transcription (time annotated string and fret position)

(iii) Performed chords (time-annotated chords performed by the player)

2.2. Neural Networks for Musical Chord Recognition. The authors of [15, 16] consider the complex problem of music recognition in their article and present an effective machine learning-based approach for chord recognition using a feed-forward neural network. The method employs the pitch class profile (PCP), a well-known function vector for automatic chord recognition. While we are going to try a different approach with our model, the data used for the training of the model in this research paper is an invaluable addition to the dataset [17–19]. The dataset used in this paper consists of guitar chords recorded both in a studio (anechoic chamber) and in noisy environments. The dataset contains 2000 chords split into ten classes, giving up to 200 chords per chord type. The earlier mentioned ten classes are the most commonly used chords, which are mentioned in Table 3.

Since these are used commonly while playing the guitar, they are bound to show up more when transcribing an audio sample. Having 200 extra recordings for each of these commonly used chords will provide us with that desirable bias in our dataset eventually resulting with our model being more capable of recognizing these chords.

The files in this dataset are stored in raw WAV 16 bits mono 44100 Hz format, which is an uncompressed audio format ensuring that we get the most out of each recording in the dataset. The dataset used in this research was created by the MONTEFIORE RESEARCH GROUP of University of Liège - Montefiore Institute.

3. CNN Model

Deep neural networks and deep learning are both strong and well-known algorithms. The meticulous design of the neural network architecture is a big part of their success. We will utilise a convolutional neural network because we will be dealing with photos. In deep learning, a convolutional neural

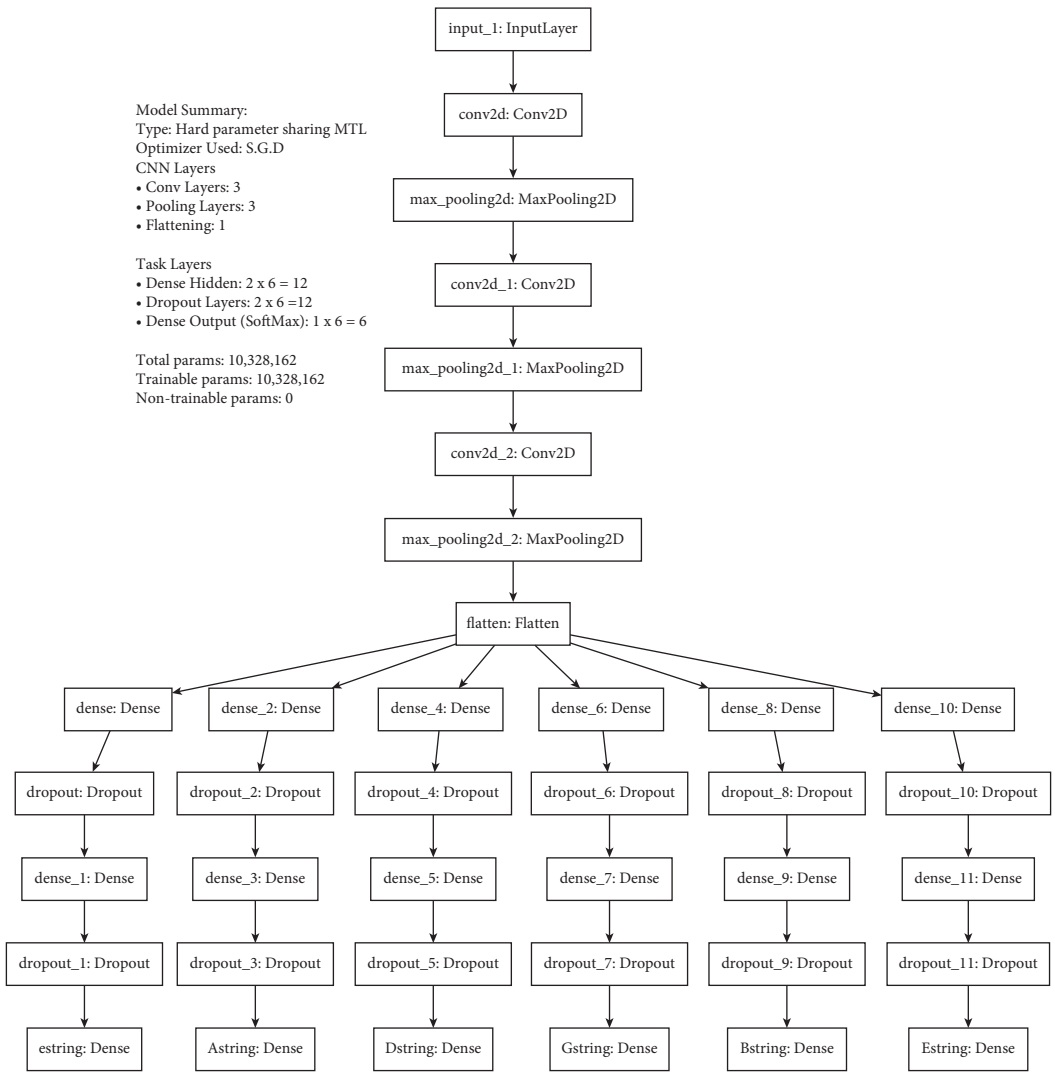

Figure 3: CNN layered architecture.

network (CNN or ConvNet) is a type of deep neural network that is most typically used to analyse visual imagery. Convolutional neural networks typically include three layers: convolution, pooling, and nonlinearity [20–23].

3.1. Model Architecture. The convolution layer is used to extract spatial features, which are then subsampled in the pooling layer using the spatial average (or max) function, and finally, the retrieved features from the previous two layers are understood by the fully connected layers with nonlinear activations (such as ReLU, tanh, or sigmoid). When applying a softmax function, this usually results in a single output node with a single value or maybe numerous output nodes resulting in a vector. [24–26].

But we have a very specific requirement concerning the output, which is that it is supposed to give out a string-fret combination, represented by means of a 6×19 matrix, where 6 represents the 6 strings of a traditional guitar and 19 represents the 18 frets and an extra place when no note is played using that string. To yield this kind of output from a deep learning model, we choose to opt for an architecture in DNN called the multitask learning [27–29]. Multitask learning has proven to be effective in a variety of machine learning applications, including natural language processing and speech recognition, as well as computer vision and drug discovery. The MTL has been referred to by a variety of titles, including joint learning, learning to learn, and learning with auxiliary tasks. In general, you are executing multitask learning (as opposed to single-task learning) as soon as you find yourself improving more than one loss function, which is precisely what we want to achieve. We can have an architecture that branches out into six endpoints, each of which produces a vector of size 19. We are expecting that each branch will become an expert at distinguishing sound from a certain string. The characteristics recovered by a common convolutional layer will be used by all six branches [30–32]. This can be better understood by looking at the summarized plot of our model given in Figure 3.

Here, each of the leaf nodes is of size 19, meaning that each of them outputs a vector of size 19 using a softmax function. The softmax function is a function that turns a

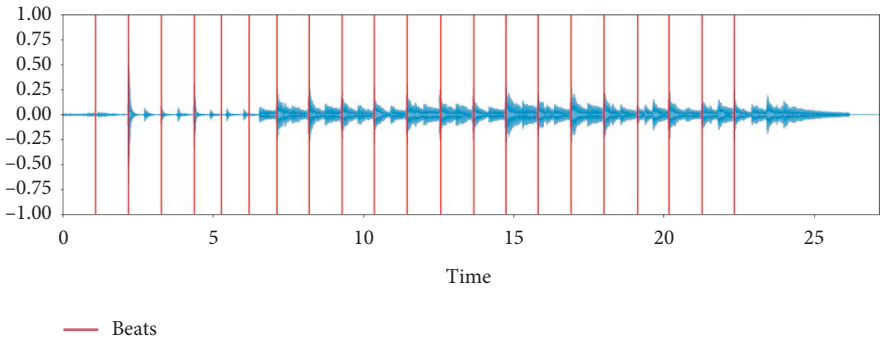

Figure 4: Waveform visualization.

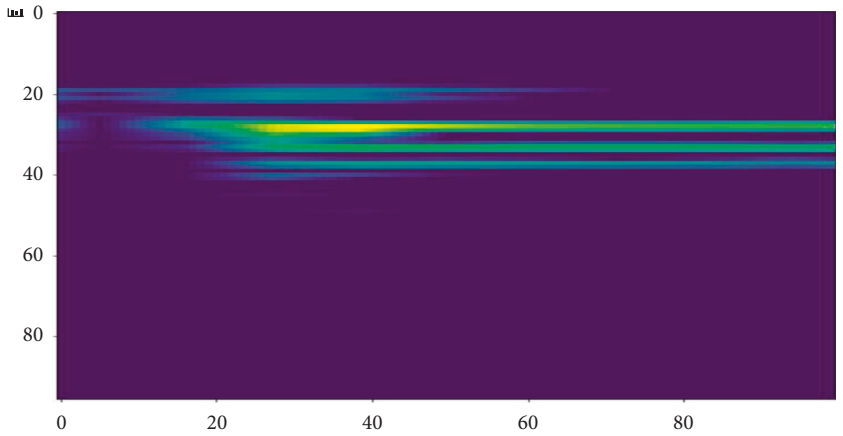

Figure 5: Spectrogram from audio samples.

vector of K real values into a vector of K real values that sum to 1, and the one with the highest value is assigned the value "1" and the others are assigned "0." Combining each of these vectors of size 19 from each of the 6 branches results in a string-fret combination of the entire fret-board of the guitar to play the sound that was represented in the input image (spectrogram), while the shared layers of repeated convolution-pooling chain are inspired from the LeNet architecture [6] because the LeNet architecture is an excellent architecture to start with, and its simplicity is unmatched.

3.2. Data Preprocessing. Preprocessing and data preparation are two of the most crucial and difficult processes in any machine learning project. The reason for this is because each dataset is unique and tailored to the project. Since we will be working with two datasets, we will need to make sure they are both in a consistent, logical, and compact manner so that the model can analyse them.

3.2.1. Beat Tracking. The phrase "audio beat tracking" refers to detecting the points in an audio recording when a human listener is likely to tap their foot in rhythm with the music. The "beat-synchronous" analysis of music is made possible by audio beat tracking. This aids us in anticipating when a guitar string will be plucked.

The plot [Figure 4] shows what detected beats look like when visualized over a waveform of an audio. These beats become extremely important as we are generating spectrograms for a window of only 0.2 seconds. Hence, we need to make sure that we capture the sound of the plucking of string(s) within that window of time.

3.2.2. Creating Spectrograms from Audio Samples. Since we will be utilising the constant-Q transform to build spectrograms, we will talk about how to use it in Python. Using the libROSA module, the constant-Q transform may be readily applied to audio files in Python. We may construct the input pictures required to train the CNN by running each audio file in the GuitarSet dataset from a defined start time (start) to a specified duration (dur) and saving the result as an image. [33–35]. For this work, dur was set to 0.2 seconds and start was set to increase from zero to the length of each audio file by the set duration, which can yield as shown in Figure 5.

Note that when being used as input images to the CNN, the color scheme was first converted to the gray scale.

3.2.3. Dataset from the Montefiore Research Group. While the GuitarSet dataset provides a well-structured time-based string-fret combination annotation for each audio sample, it is not the case with the second dataset containing

Mode		LastWriteTime	Lenght	Name
d-----	25-02	05:56 PM		a
d-----	25-02	05:56 PM		am
d-----	25-02	05:56 PM		bm
d-----	25-02	05:56 PM		c
d-----	25-02	05:56 PM		d
d-----	25-02	05:56 PM		dm
d-----	25-02	05:56 PM		e
d-----	25-02	05:56 PM		em
d-----	25-02	05:57 PM		f
d-----	25-02	05:57 PM		g

Figure 6: Sample data - GuitarSet

common chords. In fact, the only form of labelling it had was the way it was stored.

As it can be seen, the directory of the dataset was its only labelling [Figure 6]. Each folder is named after a chord containing 200 sample audios of the same. Hence, labelling of this dataset was essentially done by systematic traversal of audio samples using a python script, i.e., a predefined string-fret combination was assigned to each audio sample based on the directory it was present in (which also represented the chord being played in the samples present in the directory). The exact format of how the string-fret combination is stored is discussed later in this section.

3.2.4. GuitarSet. For the CNN model to change its estimates, each constant-Q transform picture must have a solution. Fortunately, the GuitarSet dataset contains all of the notes played for each of the 6 strings as MIDI values (pitch is commonly indicated by MIDI number in electronic music), as well as the time each note begins in the recording and the duration of the note for each audio file. This allows us to extract the string-fret combination by simply recording the MIDI values arriving on each string for a set period of time (in our example, 0.2 seconds) and labelling the spectrogram made for the audio of that period with the retrieved string-fret combinations.

First, let us take a look at a matrix (6, 18) of MIDI values, which represents the six strings and 18 frets of a guitar:

$$\begin{bmatrix} [40\ 41\ 42\ 43\ 44\ 45\ 46\ 47\ 48\ 49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57] \\ [45\ 46\ 47\ 48\ 49\ 50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60\ 61\ 62] \\ [50\ 51\ 52\ 53\ 54\ 55\ 56\ 57\ 58\ 59\ 60\ 61\ 62\ 63\ 64\ 65\ 66\ 67] \\ [55\ 56\ 57\ 58\ 59\ 60\ 61\ 62\ 63\ 64\ 65\ 66\ 67\ 68\ 69\ 70\ 71\ 72] \\ [59\ 60\ 61\ 62\ 63\ 64\ 65\ 66\ 67\ 68\ 69\ 70\ 71\ 72\ 73\ 74\ 75\ 76] \\ [64\ 65\ 66\ 67\ 68\ 69\ 70\ 71\ 72\ 73\ 74\ 75\ 76\ 77\ 78\ 79\ 80\ 81] \end{bmatrix}. \tag{2}$$

Now, an image of the guitar frets [Figure 7] with corresponding notes labelled in each string-fret combination: Notice that each note within a particular octave is associated with a unique MIDI value.

3.2.5. Final Format of the Dataset. In this case, we were interested in time annotated string-fret combinations for the given audio. To make it understandable by our CNN model, we want the audio to be in the form of a spectrogram for a 0.2-second audio piece. Keeping all of that in mind, we end up with a format dataset.

(1) X = flattened spectrogram image (96×9) of a 0.2 second audio sample [Size = 864] (which will later be reshaped before it reaches the input layer of the model)

(2) Ye = Vector representing frets of "e" string

(3) YA = Vector representing frets of "A" string

(4) YD = Vector representing frets of "D" string

(5) YG = Vector representing frets of "G" string

(6) YB = Vector representing frets of "B" string

(7) YE = Vector representing frets of "E" string

All the Y's are one hot encoded vector of size 19 where the index with value one represents fret pressed. The 0th index indicates whether or not the string is being played, the 1st index indicates whether or not the open string is being played, and the third through nineteenth columns indicate which fret is being played, beginning with the first fret. The final size of the DataFrame is given in Figure 8.

3.2.6. Normalization. Normalization is a scaling method that shifts and rescales data to make them range between 0 and 1. Min-max scaling is another name for it. Here, the formula for normalization is as follows:

$$X' = \frac{X - X_{\min}}{X_{\max} - X_{\min}}. \tag{3}$$

It helps the performance of the model when the range of inputs are in a set known scale, as it fixates some certainty in the range of weights that we will be looking at [36, 37].

3.3. Model Training

3.3.1. Training Platform. Looking at the mammoth size of the samples in the dataset and the image processing task, it is not feasible to perform model training of this size on even an above-average spec computer. The nature of the computation is convolutional, which allows computing on the graphic processor unit (GPU), which accelerates the whole process to up to 10 times based on the GPU being used [38–40]. For these computing resources, we decided to rely on Google Collaboratory.

Google Collaboratory, or "Colab" for short, allows you to develop and run Python code in your browser with

(i) No configuration necessary

(ii) Free access to GPUs

(iii) Simple sharing

3.3.2. Optimizer. The stochastic gradient descent with momentum was the optimizer of choice in our scenario. In contrast, stochastic gradient descent (SGD) updates a parameter for each training example $x^{(i)}$ and label $y^{(i)}$:

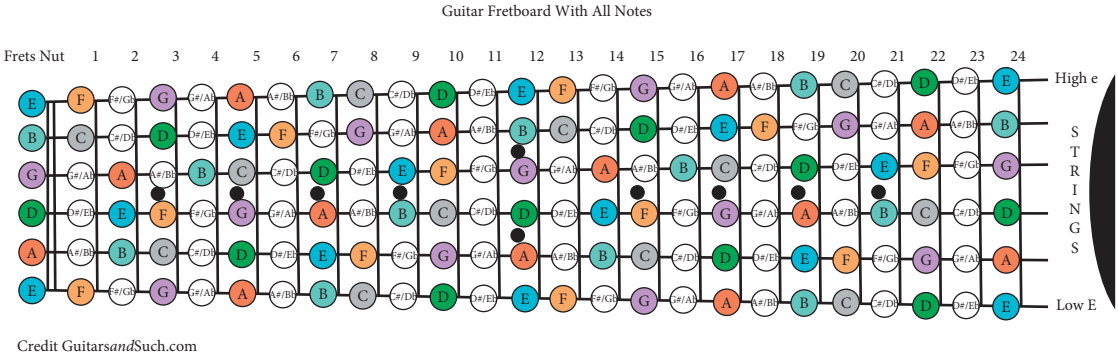

Figure 7: Guitar fret board with all notes.

	x	Y_e	Y_A	Y_D	Y_G	Y_B	Y_E
count	145176	145176	145176	145176	145176	145176	145176

Figure 8: Final size of the DataFrame.

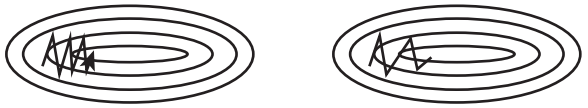

Figure 9: Stochastic gradient descent.

$$\theta = \theta - \eta \cdot \nabla_\theta J\left(\theta; x^{(i)}; y^{(i)}\right). \tag{4}$$

SGD, on the other hand, has problems navigating ravines, which are abundant near local optima and are locations where the surface slopes considerably more sharply in one dimension than in another. SGD oscillates throughout the ravine's slopes while only making halting progress towards the bottom towards the local optimum in these cases, as shown in Figure 9.

As seen in the second figure, momentum is a technique that aids in the acceleration of SGD in the desired direction while also dampening oscillations. It does this by adding a fraction γ of the update vector of the past time step to the current update vector:

$$V_t = \gamma V_{t-1} + \eta \nabla_\theta J(\theta). \tag{5}$$

Some implementations exchange the signs in the equations. The momentum term γ is usually set to 0.9 or a similar value. Essentially, when using momentum, we push a ball down a hill. The ball accumulates momentum as it rolls downhill, becoming faster and faster on the way (until it reaches its terminal velocity if there is air resistance, i.e., $\gamma < 1$). The momentum term grows for dimensions whose gradients' point is in the same directions and decreases for dimensions whose gradients change directions for our parameter updates. We get faster convergence and less oscillation as a result of this. [41, 42].

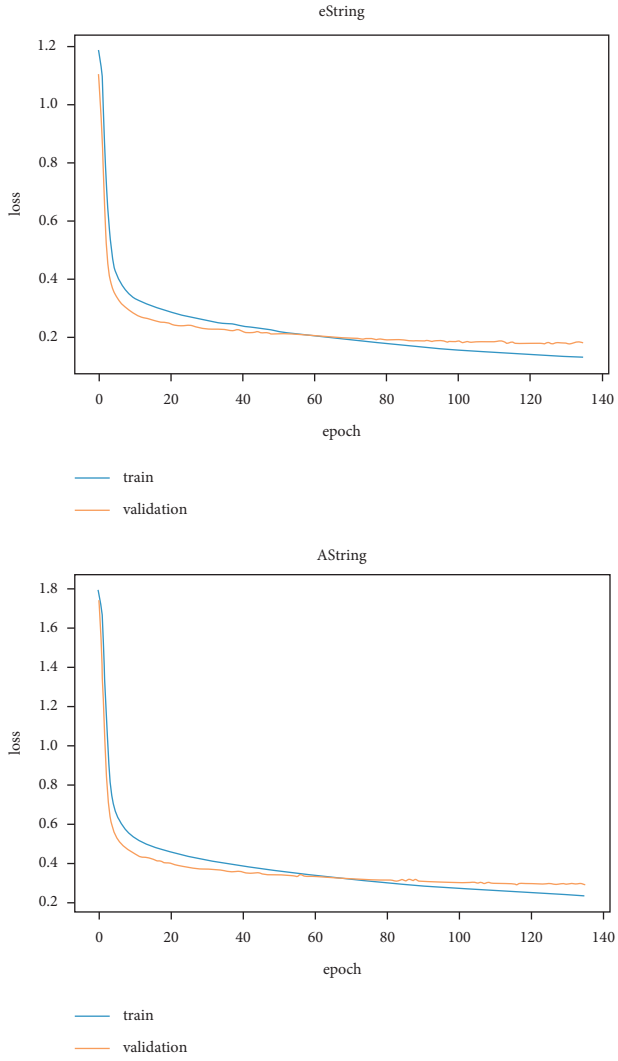

Figure 10: E and A string learning curve.

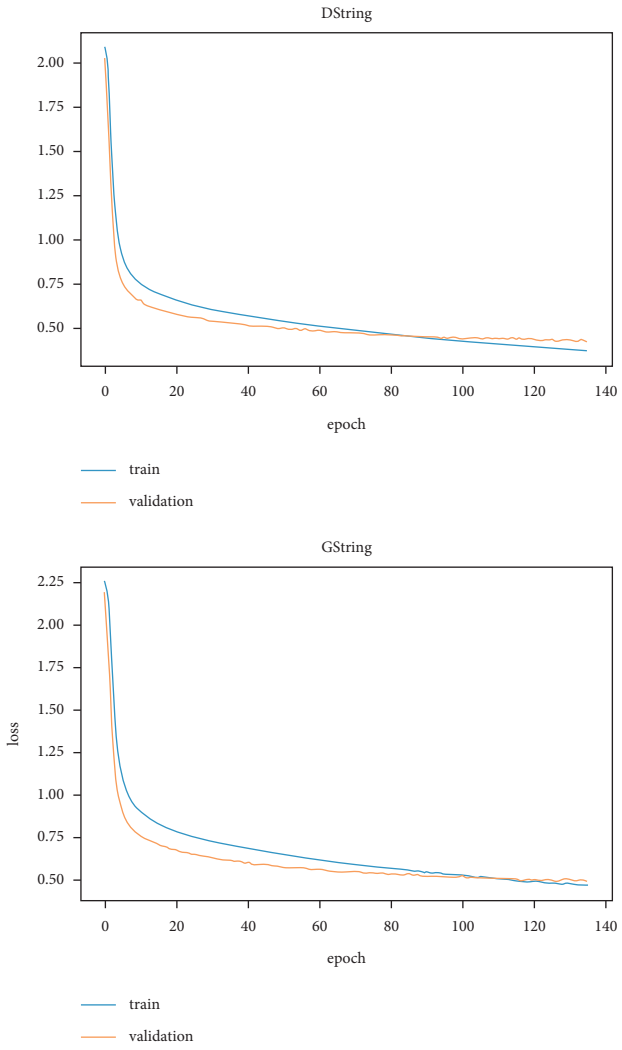

Figure 11: D and G string learning curve.

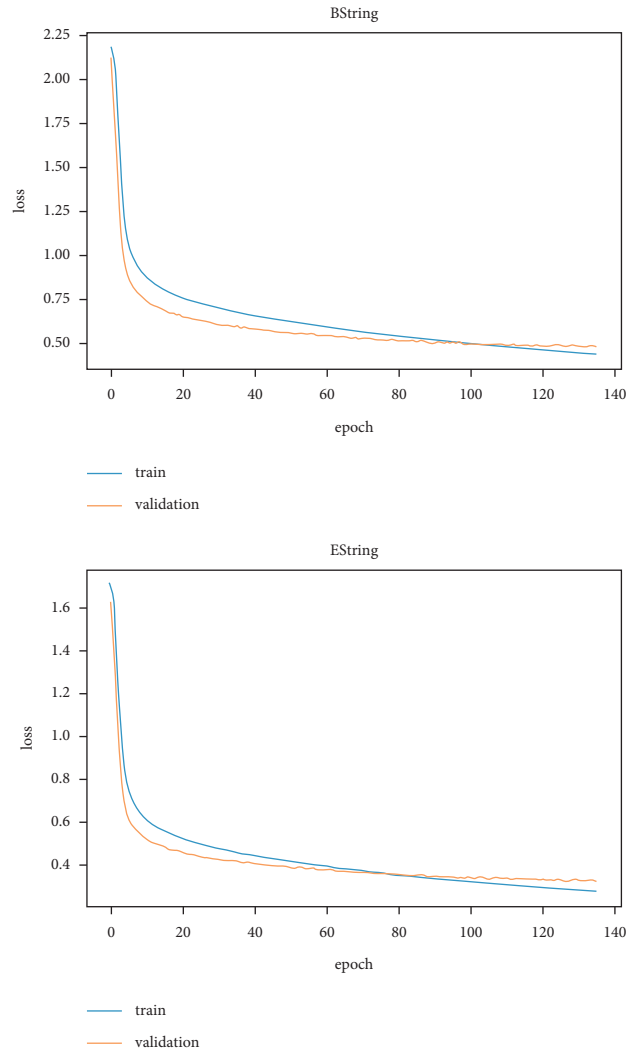

Figure 12: B and E string learning curve.

4. Result Analysis

A learning curve is a graph representing model learning performance as a function of time or experience. In machine learning, learning curves are a common diagnostic tool for algorithms that learn progressively from a training dataset. After each update during training, the model may be tested on the training dataset and a holdout validation dataset, and graphs of the measured performance can be constructed to display learning curves. We will be looking at the optimization learning curve [Figure 10], which is based on the measure used to optimise the model's parameters.

Since the model performs six tasks (six branches), each predicting an output for each of the six strings of a standard tuned guitar e, A, D, G, B, and E, we have 6 learning curves for the loss of each of the strings. This allowed us to monitor the performance of the model on each task. It is evident from the curves that the model is good at predicting results for the *e* string, while it does not do that well with *G* string as the loss for the *e* string is much less than the loss of *G* string. However, it seems like a complicated task to evaluate the models' overall performance looking at 6 learning curves

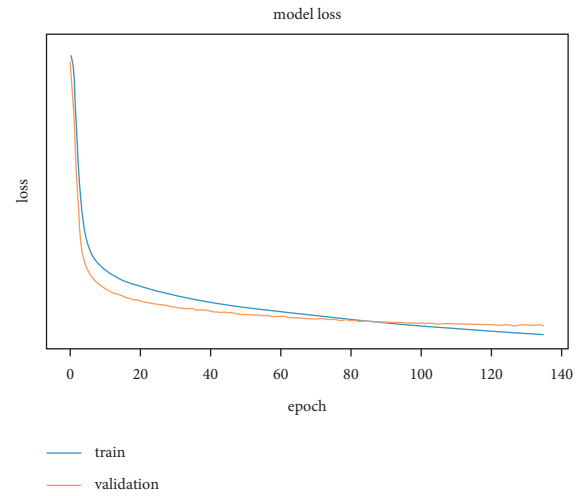

Figure 13: Model loss.

Table 4: Performance accuracy for all strings

Task	Accuracy in percentage (%)
e string performance	95.05
A string performance	91.54
D string performance	86.75
G string performance	84.00
B string performance	85.15
E string performance	89.78
Overall performance	88.71

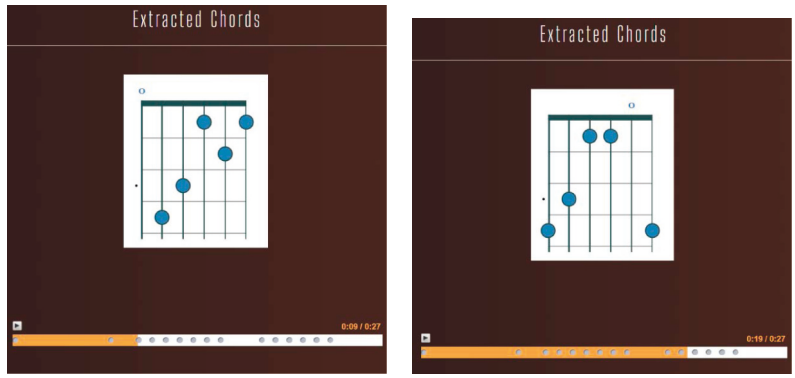

Figure 14: Sample extracted cord.

[as shown in Figures 11–13]; hence, we also plot an overall optimization learning curve of the model, which averages loss for all the six strings and summarizes the model performance with a single graph as shown in Figure 13.

It is quite evident from the above graph that the model has reached its learning saturation as the validation loss has stabilized more in terms of its rate of decay, and the training loss continues to go down with a greater rate. If we train the model any further, the model will start overfitting, meaning that it will be better on the training data and poor on unseen data; it can be said that it will start memorizing the training data, or it will not generalize its knowledge properly.

The training in the above graphs has been stopped when the validation loss was the least. Usually in these cases, we start with a random number of epochs and study the learning curve and observe at how many epochs does the model generalizes the best, and then, we start training the model again, and this time, we stop the training early at the number of epochs we observed earlier to achieve the most optimal weights. When using learning rate decay (when the learning rate decreases with each epoch, slowly decaying towards zero, the rate at which it decays depends on the total number of epochs the model has to train for), the observed optimal number of epochs varies as the number of epochs to train for changes.

The final CNN model's accuracy is summarized in Table 4. The accuracy of 88.71% is quite reasonable to be used in real life scenarios. While the proposed model was inspired by an article by Tio [7], it overcomes many of the drawbacks posed by the former attempts made towards this problem. To list a few, the proposed model has an extra feature where it detects the beats before clipping out snippets of the guitar solo. The overall accuracy was also improved by incorporating multiple datasets and by making sure that the model generalizes better than the former version by the means of data augmentation (addition of noise in the audio samples). Hence, the proposed model poses some value in the field of music information retrieval (MIR) while it also provides budding guitarists a better platform to learn from.

5. Conclusion

A method for detecting chord sequences and notes from audio recordings of solo guitar performance was presented in the study. The proposed method gives chord names and the notes on the fret-board for intervals in the input audio. The proposed model has been tweaked to reach an accuracy of 88.7%. The musical models that were constructed expressly for the guitar, as well as the acoustic models that were trained using guitar acoustic material, are responsible for the improvement. This illustrates how creating music transcription algorithms for more tightly defined scenarios may lead to considerable performance gains. Because of its accuracy, the suggested system is a suitable choice for usage by the target audience.

At this point, the model can detect the chords and notes from an audio file that has only a guitar and no other instruments, as shown in Figure 14. This can be quite restrictive, especially for the target audience as they may need to get tablatures from an actual song with various other instruments and vocals. To tackle this issue, we could add a stem separation tool to preprocess our audio file before feeding it to the model. This tool would extract specific elements from the audio file such as vocals, drums, the guitar, and the keyboard, isolating the guitar part for our model. This would be a massive upgrade as the variety of

audio files that can be used will increase dramatically and would save the user incredible amounts of time as they would not have to search for a guitar solo for the particular song/audio.

The proposed scheme could be diversified even more by expanding our target audience, which can be done by adding the option to get transcriptions for various other instruments such as the piano and drums. This would require more models with more training and testing on the different datasets for each other and the different instruments. There are many datasets for all these other instruments, and different approaches could be made if the need be. This enlarges the number of people who can use this tool.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] L. Alcabasa and N. Marcos, "Automatic guitar music transcription," in *Proceedings of the International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, pp. 197–202, IEEE, Kuala Lumpur, Malaysia, November 2012.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] J. Osmalsky, J.-J. Embrechts, M. Van Droogenbroeck, and S. . Pierard, "Neural networks for musical chords recognition," *Journées d'informatique musicale*, pp. 39–46, 2012, https://orbi.uliege.be/handle/2268/115963.

[4] D. Tio, *Automated Guitar Transcription with Deep Learning - towards Data Science*, p. 12, 2021.

[5] C. Schörkhuber and A. Klapuri, "Constant-q transform toolbox for music processing," in *7th Sound and Music Computing Conference*pp. 3–64, Barcelona, Spain, 2010.

[6] A. M. Barbancho, A. Klapuri, L. J. Tardon, and I. Barbancho, "Automatic transcription of guitar chords and fingering from audio," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, no. 3, pp. 915–921, 2012.

[7] D. Tio, *Audio to Guitar Tab with Deep Learning*, https://arxiv.org/abs/2008.01431, 2019.

[8] I. Barbancho, L. J. Tardon, S. Sammartino, and A. M. Barbancho, "Inharmonicity-based method for the automatic generation of guitar tablature," *IEEE Transactions on Audio Speech and Language Processing*, vol. 20, no. 6, pp. 1857–1868, 2012.

[9] G. Burlet and I. Fujinaga, "Robotaba guitar tablature transcription framework," *ISMIR*, pp. 517–522, 2013, https://ismir2013.ismir.net/wp-content/uploads/2013/09/217_Paper.pdf.

[10] K. Yazawa, K. Itoyama, and H. G. Okuno, "Automatic transcription of guitar tablature from audio signals in accordance with player's proficiency," in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3122–3126, IEEE, Florence, Italy, May 2014.

[11] E. J. Humphrey and J. P. Bello, "From music audio to chord tablature: teaching deep convolutional networks toplay guitar," in *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6974–6978, IEEE, Florence, Italy, May 2014.

[12] C. Kehling, J. Abeßer, C. Dittmar, and G. Schuller, "Automatic tablature transcription of electric guitar recordings by estimation of score-and instrument-related parameters," in *Proceedings of the 17th International Conference on Digital Audio Effects (DAFx-14)*, pp. 219–226, Erlangen, Germany, September 2014.

[13] M. McVicar, S. Fukayama, and M. Goto, "Autoguitartab: computer-aided composition of rhythm and lead guitar parts in the tablature space," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 7, pp. 1105–1117, 2015.

[14] Joao Victor Ramos, André Stylianos Ramos, C. N. Silla, and Danilo Sipoli Sanches, "An evaluation of different evolutionary approaches applied in the process of automatic transcription of music scores into tablatures," in *Proceedings of the 2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pp. 663–669, IEEE, San Jose, CA, USA, November 2016.

[15] Q. Xi, R. M. Bittner, J. Pauwels, X. Ye, and Juan Pablo Bello, "Guitarset: a dataset for guitar transcription," in *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR*, pp. 453–460, 2018.

[16] S. Ruder, "An Overview of Multi-Task Learning for Deep Learning," 2017, https://doi.org/10.48550/arXiv.1706.05098.

[17] T. Ooaku, T. D. Linh, M. Arai, T. Maekawa, and K. Mizutani, "Guitar chord recognition based on finger patterns with deep learning," in *Proceedings of the 4th International Conference on Communication and Information Processing*, pp. 54–57, Qingdao, China, November 2018.

[18] H. Mukherjee, A. Dhar, B. Paul et al., "Deep learning-based music chord family identification," in *International Conference on Intelligent Computing and Communication*, pp. 175–184, Springer, Singapore, 2019.

[19] G. Byambatsogt, L. Choimaa, and G. Koutaki, "Guitar chord sensing and recognition using multi-task learning and physical data augmentation with robotics," *Sensors*, vol. 20, no. 21, p. 6077, 2020.

[20] F. Alenezi, A. Armghan, S. N. Mohanty, R. H. Jhaveri, and P. Tiwari, "Block-greedy and cnn based underwater image dehazing for novel depth estimation and optimal ambient light," *Water*, vol. 13, no. 23, p. 3470, 2021.

[21] P. N. Srinivasu, A. K. Bhoi, R. H. Jhaveri, G. T. Reddy, and M. Bilal, "Probabilistic deep q network for real-time path planning in censorious robotic procedures using force sensors," *Journal of Real-Time Image Processing*, vol. 18, no. 5, pp. 1773–1785, 2021.

[22] A. Patel and J. Shah, "Sensor-based activity recognition in the context of ambient assisted living systems: a review," *Journal of Ambient Intelligence and Smart Environments*, vol. 11, no. 4, pp. 301–322, 2019.

[23] A. Patel and J. Shah, "Towards enhancing the health standards of elderly: role of ambient sensors and user perspective," *International Journal of Engineering Systems Modelling and Simulation*, vol. 13, no. 1, pp. 96–110, 2022.

[24] A. Wright, E.-P. Damskägg, L. Juvela, and V. Välimäki, "Real-time guitar amplifier emulation with deep learning," *Applied Sciences*, vol. 10, no. 3, p. 766, 2020.

[25] H. V. Koops, W. B. de Haas, J. Bransen, and A. Volk, "Automatic chord label personalization through deep learning of

shared harmonic interval profiles," *Neural Computing & Applications*, vol. 32, no. 4, pp. 929–939, 2020.

[26] A. D. Patel and H. S. Jigarkumar, "Performance analysis of supervised machine learning algorithms to recognize human activity in ambient assisted living environment," in *Proceedings of the 2019 IEEE 16th India Council International Conference (INDICON)*, pp. 1–4, IEEE, Rajkot, India, December 2019.

[27] L. Alzubaidi, J. Zhang, A. J. Humaidi et al., "Review of deep learning: concepts, cnn architectures, challenges, applications, future directions," *Journal of big Data*, vol. 8, no. 1, pp. 53–74, 2021.

[28] S. Mohammed Yusuf, E. A. Adedokun, M. B. Muazu, I. J. Umoh, and A. A. I.. Rmwsaug, "Robust multi-window spectrogram augmentation approach for deep learning based speech emotion recognition," in *Proceedings of the 2021 Innovations in Intelligent Systems and Applications Conference (ASYU)*, pp. 1–6, IEEE, Elazig, Turkey, October 2021.

[29] H. Duysak, U. Ozkaya, and E. Yigit, "Determination of the amount of grain in silos with deep learning methods based on radar spectrogram data," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2021.

[30] A. S. Khan, Z. Ahmad, J. Abdullah, and F. Ahmad, "A spectrogram image-based network anomaly detection system using deep convolutional neural network," *IEEE Access*, vol. 9, pp. 87079–87093, 2021.

[31] T. Arias-Vergara, P. Klumpp, J. C. Vasquez-Correa, E. Nöth, J. R. Orozco-Arroyave, and M. Schuster, "Multi-channel spectrograms for speech processing applications using deep learning methods," *Pattern Analysis & Applications*, vol. 24, no. 2, pp. 423–431, 2021.

[32] Y. Wang, M. Zhang, RuM. Wu et al., "Silent speech decoding using spectrogram features based on neuromuscular activities," *Brain Sciences*, vol. 10, no. 7, p. 442, 2020.

[33] H. W. Loh, C. P. Ooi, E. Aydemir, T. Tuncer, S. Dogan, and U. R. Acharya, "Decision support system for major depression detection using spectrogram and convolution neural network with eeg signals," *Expert Systems*, vol. 39, no. 3, 2022.

[34] H. Chaurasiya, "Time-frequency representations: spectrogram, cochleogram and correlogram," *Procedia Computer Science*, vol. 167, pp. 1901–1910, 2020.

[35] A. Lauraitis, R. Maskeliūnas, R. Damaševičius, and T. Krilavičius, "Detection of speech impairments using cepstrum, auditory spectrogram and wavelet time scattering domain features," *IEEE Access*, vol. 8, pp. 96162–96172, 2020.

[36] Y. Jadhav and H. Mathur, "Detection of breast cancer by using various machine learning and deep learning algorithms," in *Handbook of Machine Learning for Computational Optimization*, pp. 51–70, CRC Press, Boca Raton, Florida USA, 2021.

[37] Y. Jadhav, V. Patil, and D. Parasar, "Machine learning approach to classify birds on the basis of their sound," in *Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT)*, pp. 69–73, IEEE, Coimbatore, India, February 2020.

[38] M. N. A. Tawhid, S. Siuly, H. Wang, F. Whittaker, K. Wang, and Y. Zhang, "A spectrogram image based intelligent technique for automatic detection of autism spectrum disorder from eeg," *PLoS One*, vol. 16, no. 6, 2021.

[39] Q. Zhou, J. Shan, W. Ding et al., "Cough recognition based on mel-spectrogram and convolutional neural network," *Frontiers in Robotics and AI*, vol. 8, Article ID 580080, 2021.

[40] T. Cody and P. A. Beling, "Heterogeneous transfer in deep learning for spectrogram classification in cognitive communications," in *Proceedings of the 2021 IEEE Cognitive Communications for Aerospace Applications Workshop (CCAAW)*, pp. 1–5, IEEE, Cleveland, OH, USA, June 2021.

[41] G. Liu, X. Han, L. Tian, W. Zhou, and H. Liu, "Ecg quality assessment based on hand-crafted statistics and deep-learned s-transform spectrogram features," *Computer Methods and Programs in Biomedicine*, vol. 208, Article ID 106269, 2021.

[42] F. A. Bhatti, M. J. Khan, A. Selim, and F. Paisana, "Shared spectrum monitoring using deep learning," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 4, pp. 1171–1185, 2021.