Philippine e-Journal for Applied Research and Development 9(2019), 49-57 ISSN 2449-3694 (Online)
http://pejard.slu.edu.ph/vol.9/2019.10.26.pdf

Classification of Acoustic Guitar Strum using Convolutional Neural Networks and Long-Short-Term-Memory

Krishia Bello[1] and Paula Mayol[1*]

[1]*Department of Computer Science, College of Science, University of the Philippines Cebu, Gorordo Avenue, Lahug, 6000 Cebu City*

* *Corresponding author (psesplanada@up.edu.ph)*

Received, 04 June 2019; Accepted, 10 October 2019; Published, 26 October 2019

Abstract

Audio music feature recognition has been an active research area due to its various applications like music instrument classification. However, there is little literature investigating the possibility of classifying acoustic guitar strumming. This study aimed to classify acoustic guitar strumming in downstroke and upstroke style category, utilizing two neural network based classifier models. Each audio data were manually segmented per strum, and subjected to noise reduction through audio signal emphasis, downsampling, and MFCC conversion, before being fed to the designed neural network models. Convolutional Neural Network (CNN) and Long-Short-Term-Memory (LSTM) Recurrent Neural Network were separately used to classify the guitar strums. Results showed that CNN acquired a higher performance than LSTM with 91% model accuracy and a 19% model loss, while the LSTM model resulted in an 88% model accuracy and a 26% model loss. This indicates that CNN would be the preferred neural network model to utilize for classifying guitar strums. Future works may explore classification of other guitar types and the extraction of guitar strums in an audio file that includes other musical instruments.

Keywords: *audio music feature recognition, guitar strumming style, guitar strum classifier*

Introduction

The field of computing has made its way to the music industry innovation, from automation of musical instrument extractions to single note detection and generating musical notes. For guitar music, chord detection has been done using k-Nearest Neighbor Classifier [1]. Guitar tablatures were also generated using two approaches on Neural Networks [2]. Moreover, a wristwatch-like device was developed to determine how a player is strumming the guitar, evaluating the strumming action to quantify the musical aspects of guitar playing such as the timing and the strength of notes [3]. However, this device cannot determine if the player's guitar strum is upward or downward. In fact, to date, there is no system yet that can determine the upward or downward movement of a guitar player's strum.

Strumming patterns are difficult to comprehend and more challenging to teach. This is because strumming patterns are based on rhythm, which is better understood when absorbed intuitively and not only in an academic sense [4]. Guitar player novices can hardly catch a song's strumming pattern just by listening to the song. Moreover, strumming patterns are seldom present in music sheets. Thus, this points out to the need for a guitar strumming pattern tablature that specifically differentiates upward and downward strokes.

To produce a guitar strumming pattern tablature, a guitar's strumming pattern from a song can be examined and annotated via automation, which can potentially be done by heuristic-based algorithm approaches. Studies made on extraction and recognition of different musical instruments paved the way for other research in a deeper music element hierarchy, such as the classification of different musical concepts over its much diverse musical category. As there are already a lot of music notations like music instrument extraction and identification [5, 6], note detection [7], chord detection [1], and time signature classification [8], the classification of guitar strums will make a good addition as it enables deeper music notation through computing. In the long run, it will eventually pave the way to easily generate the guitar strumming pattern for a song.

This study aims to develop a method for processing audio data as input to an acoustic guitar strum classifier that will determine if strumming is in an upward or downward movement. The study will utilize two neural network based classifier models (namely, CNN and LSTM) to build the guitar strum classifier, which will aid guitar player novices in learning the strumming tablature. As both CNN and LSTM are considered as powerful neural networks with classifying capabilities, comparisons between the two models will be done in order to evaluate and determine which of them would fare better in terms of classifying correctly the guitar's strum.

Materials and Methods

The framework followed is divided into four parts: (1) data gathering, (2) cleaning and preprocessing, (3) training and validation, and (4) testing. Parts 1 and 2 consist of manually recording guitar strum patterns, then feeding the processed data into the two classifier models. Parts 3 and 4 train and verify the accuracy of the model's ability to classify guitar strums. Figure 1 shows the study's research framework, which

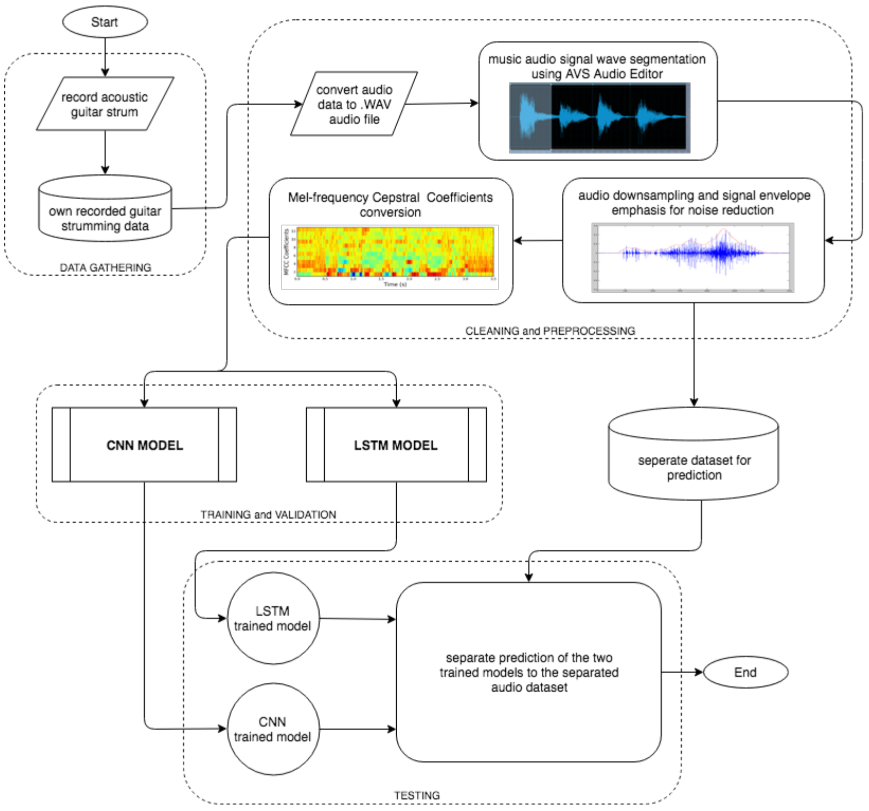
Start
record acoustic guitar strum
own recorded guitar strumming data
DATA GATHERING
convert audio data to .WAV audio file
music audio signal wave segmentation using AVS Audio Editor
Mel-frequency Cepstral Coefficients conversion
audio downsampling and signal envelope emphasis for noise reduction
CLEANING and PREPROCESSING
CNN MODEL
LSTM MODEL
TRAINING and VALIDATION
seperate dataset for prediction
LSTM trained model
CNN trained model
separate prediction of the two trained models to the separated audio dataset
End
TESTING

Figure 1. Research Framework

served as the workflow in building the guitar strum classifier.

Data Gathering, Cleaning, and Preprocessing

The audio data needed for the guitar strum classifier were obtained by manually recording guitar strumming patterns (saved as .mp3) using an acoustic guitar that was connected directly to a computer by an audio jack adapter. Guitar chords that were included in gathering the strumming data included all universal chords from *A* to *E*, with the minor chords of *A*, *B*, *D*, *E*, and the power chords from *A* to *F* with the open chord E (open strumming). These audio files were then manually segmented per strum using AVS Audio Editor, and was also converted to .wav form upon saving the file.

The data collection consisted of 2200 audio files, with no more than two seconds in length for each data, which were separated in the ratio of 2040:160 for the training and testing phase of the study, respectively. Figure 2 shows a pie chart of the class with percentage representing the mean of the overall audio length of the downstrum and upstrum classes.

Audio data were cleaned by noise reduction through signal envelope emphasis. The envelope of a signal shows the important part of the signal, so getting the envelope of a signal will also get rid of the noise in the audio data. Furthermore, the audio data were downsampled, decreasing the sampling rate from 4400 Hz to 1600 Hz. Figure 3 shows the audio data before and after envelope emphasis and downsampling.

MFCC conversion

Random features were built from the overall sample and finally converted to MFCC with parameters, 13 as number of cepstrum, 26 as number of filters in the filterbank, and 512 as the FFT size. Conversion to MFCC returns a numpy array of size (number of frames by number of cepstrum) containing features, with each row holding 1 feature vector. Building random features outputs the X matrix (samples in mfcc), and the y matrix (array of labels) that are to be fed into the neural network models. Figure 4 shows a sample of an audio data converted to MFCC.

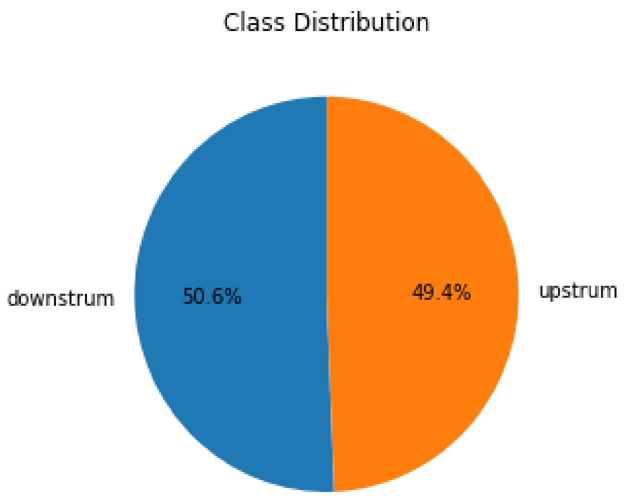

Figure 2. Class Distribution of the Dataset

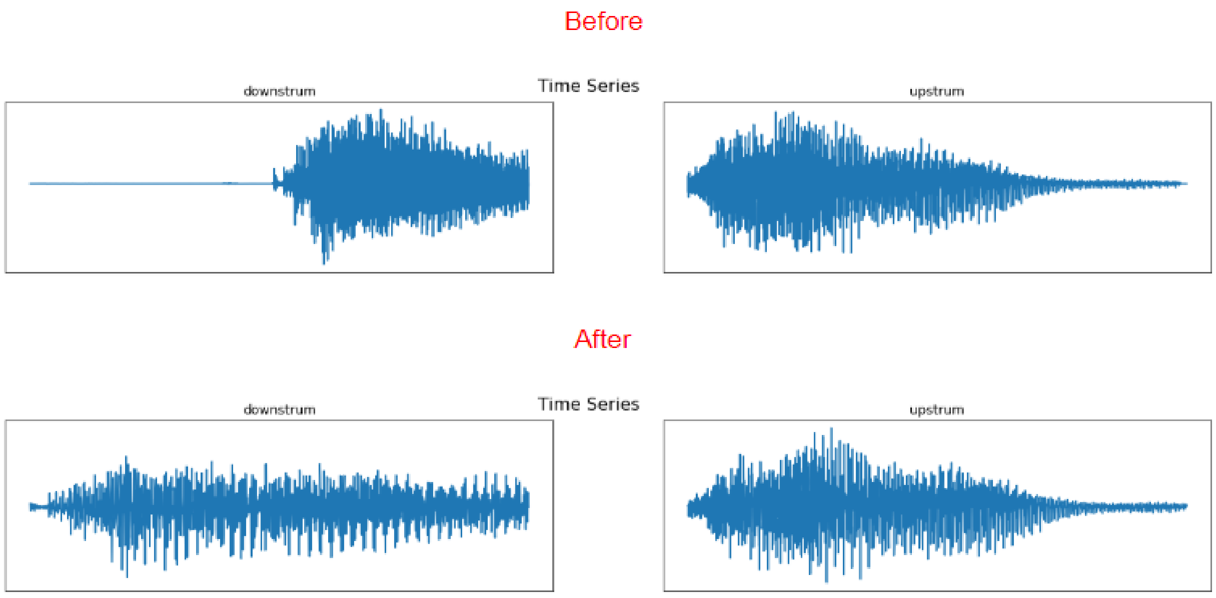

Figure 3. Audio data before and after signal envelope emphasis and downsampling

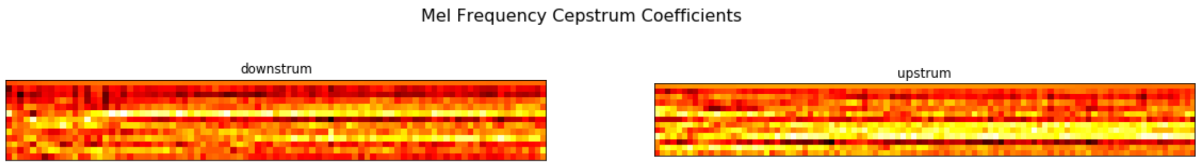

Figure 4. Mel-frequency Cepstral Coefficients audio data

Network Training and Validation

The LSTM recurrent neural network was built using the parameters as shown in Figure 5. A sequential model which is a linear stack of layers is used. The first two layers are LSTM layers with 128 memory units that returns sequences. This is done to ensure that the next layer receives sequences and not just randomly scattered data. A dropout layer is applied after the LSTM layers to avoid overfitting of the model. In the middle are four time-distributed and fully connected dense layers with decreasing dimensionalities, all having Rectified Linear Activation or “relu” as the activation function. Finally, after flattening, the network used “softmax” as activation in the dense layer that outputs 2 dimension space for the two-class classification. In compiling the model, Adaptive Moment Estimation (Adam) is used as optimizer, and categorical cross entropy is used for the loss function.

Figure 6 shows the network’s structure for CNN. This model is also sequential and starts with four Conv2D layers with increasing number of nodes per layer, all having “relu” as activation function. Next is a two-dimensional max-pooling layer for downscaling. A dropout layer is then added to control overfitting. Finally, after flattening, three dense layers, with two having “relu” as activation, and the last output dense layer with “softmax” activation function for the two-class classification, completes the CNN network model. As with the LSMT model compilation, “Adam” is used as optimizer, and categorical cross entropy is also used for the loss function.

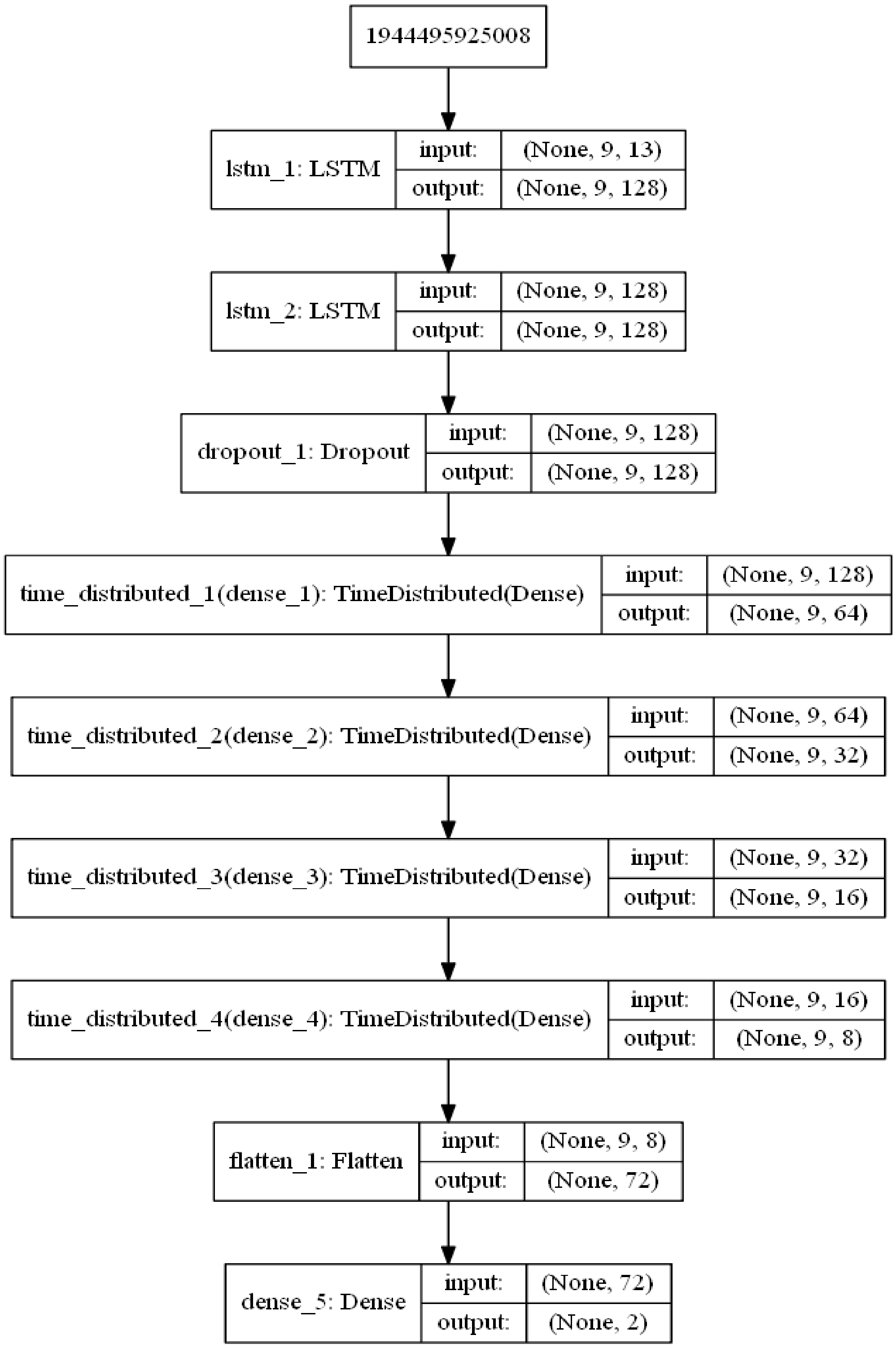

Figure 5. LSTM Network Structure

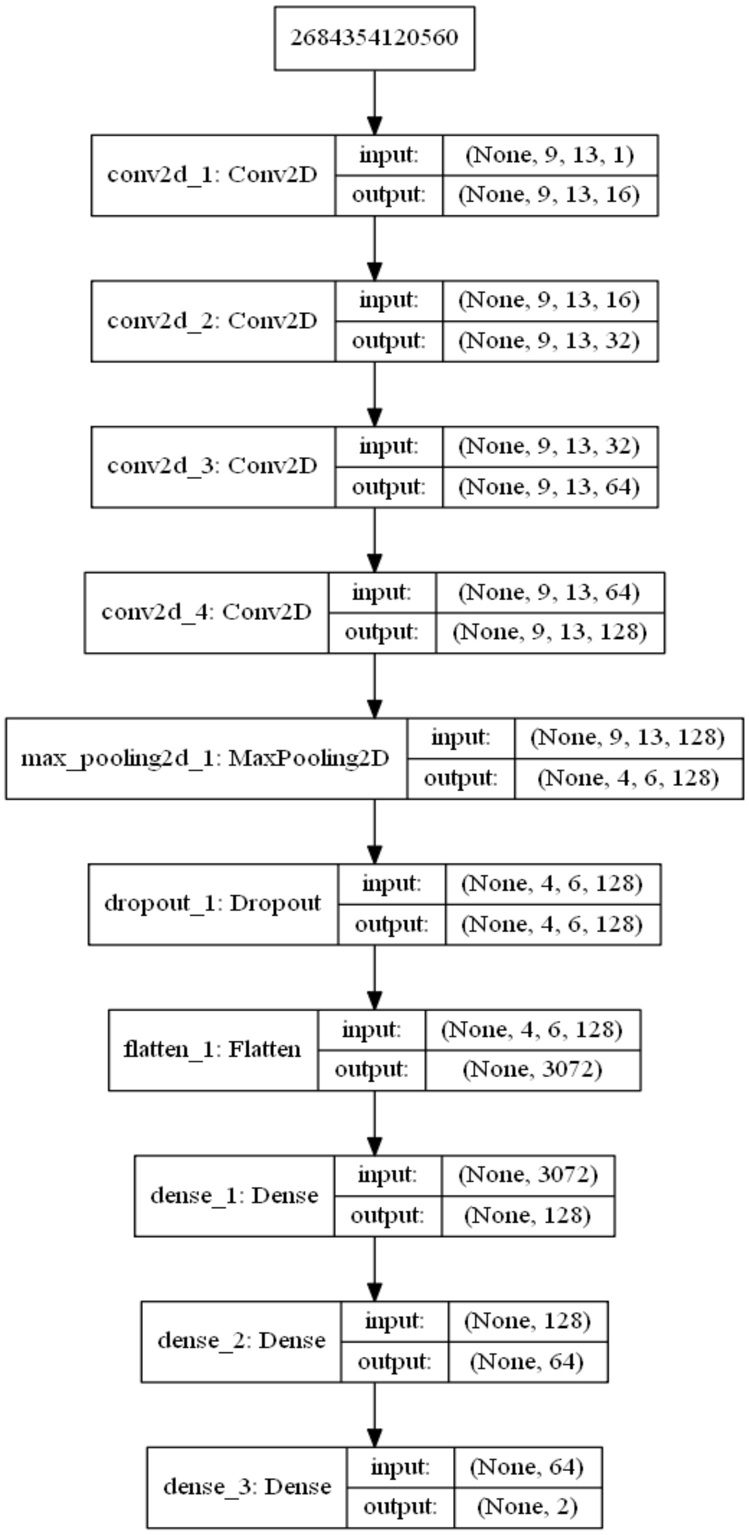

Figure 6. CNN Network Structure

Finally, to fit the created models for training and validation, validation_split with value 0.1 was assigned. This means that the model will automatically get 10% of the overall data (starting from the bottom data) and set it aside for validation. The model was trained over 65 epochs, with batch size of 32.

Results and Discussion

The classifier models were evaluated by running testing datasets and comparing the results. The separated audio data for prediction that was cleaned and preprocessed the same way as the training data were fed into the trained model for prediction. Accuracy score was then obtained both from CNN and the LSTM model upon prediction.

CNN Model Training and Validation

From the training accuracy plot of CNN model (Figure 7), 65 epochs might just be the right stopping point as the trend for accuracy on both datasets is observed to be increasing slowly during the last epochs. It can also be observed that the model has not yet over-learned the training dataset, showing comparable skill on both datasets. This CNN model obtained an accuracy of over 91%.

From the training loss plot of the CNN model, it can be said that the model has comparable performance on both train and validation datasets (labeled test). If these parallel plots start to depart consistently, it might be a sign to stop training at an earlier epoch. But 65 epochs still showed good performance, having only 19% as training loss.

LSTM Model Training and Validation

The training accuracy plot of the LSTM model (Figure 8) is not that smooth throughout the 65 epochs, but it is also not that rough getting around to over 88% accuracy. It can also be said that the training has not yet over-learned the dataset, having also comparable skill on both the training and testing datasets.

As for the training loss plot of the LSTM model, the training also showed comparable performance on both the train and validation of the labeled test. It can also be considered to stop the training around the 55th epoch because it can be seen that the parallel lines started to depart consistently. The LSTM model got around with 26% training loss.

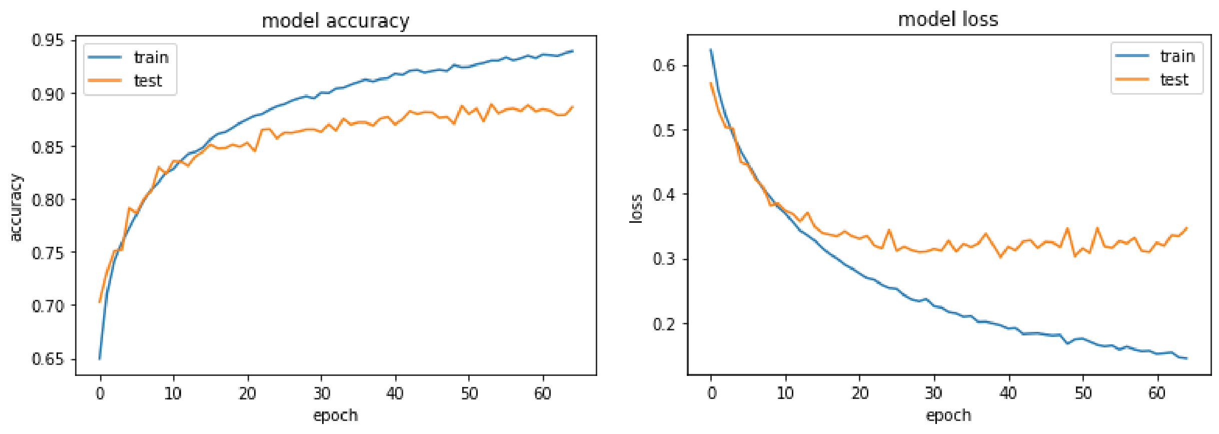

Figure 7. CNN Model Training Accuracy and Loss

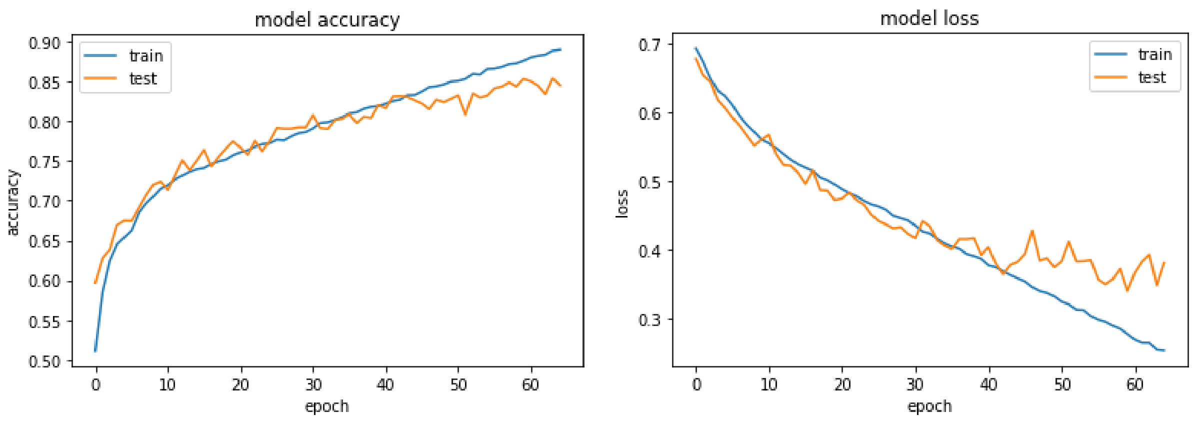

Figure 8. LSTM Model Training Accuracy and Loss

CNN and LSTM Prediction

From the separate prediction of the trained models table, out of the 160 data to be predicted, the predictions arrived at a 72.5% and 70% accuracy score for the CNN model and LSTM model, respectively (Table 1). It shows a small difference for the two models, but it still reflects the training accuracy for both models, which was 91% for CNN and 88% for LSTM.

No. of Data	CNN Result	LSTM Result
160	116 correctly predicted	112 correctly predicted
Accuracy score	72.5%	70%

Table 1. Separate Prediction of Trained Models Result

The main possible reason for the number of misclassified guitar strum data is the insufficiency of data for training. Because it was observed that for some guitar chords, especially for those that were misclassified, the classifier made only a small significant difference in its class probabilities for being a downstrum or an upstrum. More data for training is expected to give out more audio features that will eliminate the ambiguity of the audio features of a downstrum and an upstrum, given that these data are taken from a wide variety of guitar chords.

Model Training and Validation Time Duration

Model	Runtime
CNN	18:46:62
LSTM	43:38:11

Table 2. Model Training and Validation Runtime

With 65 epochs, the overall model training and validation of the CNN went to over 18 minutes, 46 seconds, and 62 milliseconds. This training time is way much faster than the LSTM that went over to 43 minutes, 38 seconds, and 11 milliseconds (Table 2). It should also be noted that the GPU card of the computer were used in the overall process of this study. Although the time complexity of a neural network depends on how it is modeled, CNN was already expected to perform faster since the computations in CNN can happen in parallel. On the other hand, LSTM performs sequential processing, since the subsequent memory nodes depend on previous memory node units. CNN proved to be much faster by design, which was reflected in this study.

Conclusion

The main objective of this study was to develop a mechanism for processing audio data as input to a guitar strum classifier, specifically noting whether it is in upward or downward stroke. To achieve this, two neural network-based models were trained from manually recorded guitar strum patterns, adopting a supervised learning approach to implement the classifier. After being subjected to several tests, both the Convolutional Neural Network model and the LSTM Recurrent Neural Network model were able to identify and correctly classify an acoustic guitar strum. For separate data prediction, CNN model obtained a higher score than LSTM. The CNN model performed better because aside from being much faster in design than the LSTM model, CNN has always been the preferred model among all the other neural network architectures to work images as input data. This suggests that in terms of classifying guitar strum audio data (represented in this study as an image in the form of MFCC audio), CNN would be the more preferred neural network based model to use

Future works should include the extraction of the guitar strumming from the file before processing the data for classification. Moreover, applying the strumming of other guitar types, such as electric and nylon-stringed guitars, will expand the scope of this study. Considering other hyperparameters in building the Neural Networks might also be done to enhance the performance of the models as well as to achieve better results. Finally, developing an actual application that could aid guitar beginners in determining the guitar strum of a given audio file will prove useful. It would also be beneficial if this can provide an interface wherein users can input any song, and the application system will then generate the guitar strumming pattern/s of the whole song.

References

[1] Mazhar, F. (2011). Automatic Guitar Chord Detection. Master's Thesis, Tampere University of Technology, Tampere, Finland. Retrieved from https://trepo.tuni.fi/bitstream/handle/123456789/20923/mazhar.pdf?sequence=7&isAllowed=y

[2] Mistler, E. (2017). Generating Guitar Tablatures with Neural Networks. Master's Thesis, University of Edinburgh, Edinburgh, Scotland. Retrieved from https://pdfs.semanticscholar.org/7065/897be10f94edb533ad8b98364372a85106da.pdf?_ga=2.81674903.1404172709.1544554413-1380252528.1544554413

[3] Matsushita, S. & Iwase, D. (2013). Detecting Strumming Action while playing Guitar. In *Proceedings of the 2013 International Symposium on Wearable Computers,* 145-146. ACM. Retrieved from: https://dl.acm.org/citation.cfm?doid=2493988.2494345

[4] Kittleberger, B. (2018, March 02). Acoustic Guitar Strumming Patterns. Retrieved from https://www.guitarworld.com/acoustic-nation/acoustic-guitar-strumming-patterns

[5] Uhlich, S., Giron, F., & Mitsufuji, Y. (2015, April). Deep Neural Network Based Instrument Extraction from Music. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2135-2139. IEEE. Retrieved from https://ieeexplore.ieee.org/abstract/document/7178348

[6] McKay, C., & Fujinaga, I. (2005, March). Automatic Music Classification and the Importance of Instrument Identification. In *Proceedings of the Conference on Interdisciplinary Musicology*, 1-10. Retrieved from https://www.researchgate.net/profile/Ichiro_Fujinaga/publication/228351016_Automatic_music_classification_and_the_importance_of_instrument_identification/links/09e4150a2948d49dc2000000.pdf

[7] Fala, J. M., Keshap, A. K., Doerning, M. S., & Barbeau, J. T. (1991). *U.S. Patent No. 4,991,488.* Washington, DC: U.S. Patent and Trademark Office.

[8] Dieleman, S., Brakel, P., & Schrauwen, B. (2011). Audio-based Music Classification with a Pretrained Convolutional Network. In *12th International Society for Music Information Retrieval Conference (ISMIR-2011),* 669-674. University of Miami. Retrieved from https://biblio.ugent.be/publication/1989534/file/6741798.pdf