Visual recognition of guitar chords using neural networks

Author: Albert Mitjans Coma
*Facultat de Física, Universitat de Barcelona, Diagonal 645, 08028 Barcelona, Spain.**

Advisor: Artur Carnicer

Abstract: In this paper, we use deep learning to study high-level features for guitar chord detection. In particular, the goal of this project is to build a neural network capable of recognizing finger patterns on the frets of a guitar. Given an input image, the network is able to identify fingers, frets, strings and the corresponding chord. Using a 2-stack Hourglass network for the detection and applying a post-processing algorithm to its corresponding output heatmaps, a 97% accuracy on the detection of chords of 205 different images is obtained.

I. INTRODUCTION

Deep learning approaches have gained significant interest in the machine learning community as a way to learn representations from large amounts of data. Deep learning has been applied successfully in various fields and studies indicate that deep learning methods are successful when applied to Music Information Retrieval (MIR) tasks [1]. Consequently, there is substantial research related to learning features from audio signals in order to detect notes and chords. However, working with audio has limitations, since any external sound or noise could potentially ruin the detection. As a result, audio-only approaches have restricted usefulness.

Image-based chord detection may help overcome this limitation, and may make feasible several applications such as musical transcription of chords played in a concert or in a music video. In addition, by combining image-based and sound-based chord detection, much more robust and reliable results may be obtained. In this paper, we want to explore if the use of object detection techniques may help us extract enough representative image features to successfully classify the chords.

Thanks to the emergence of specialized Deep Neural Networks (DNNs) such as Regions with CNN features (R-CNN)[2], reliably locating objects in an image is now realistically possible. CNNs have deep architectures with the ability to learn more complex features than the shallow networks. In addition, robust training algorithms allow these networks to learn informative object representations without the need to design features manually.

In this paper an Hourglass neural network is used to locate image features in the context of chord detection, such as the position of fingers, frets and strings. Hourglass networks have been successfully used to detect human body parts [3]. Given its structure, the network is capable of capturing information at every scale, which makes it ideal for the detection of fingers, frets and strings. Moreover the network's architecture is taken further by stacking several hourglasses end-to-end. This provides the network with a mechanism for repeated bottom-up, top-down inference allowing for reevaluation of initial estimates and features across the whole image.

In addition, a YOLO (You Only Look Once) network [4] is used to get rid of all redundant information of the image by detecting the left hand of the guitar player. YOLO is known for its predictions of bounding boxes and class probabilities directly from full images in one evaluation. Even though this model has a similar performance than other object detection models, its unified structure makes it extremely fast, which makes it ideal for the use of real-time detectors.

The paper is organized as follows. Section II explains the approach taken for this project. Section III describes the results obtained. The conclusions and future work are presented in Section IV. In addition, Section V contains a brief introduction to Deep Learning and neural networks.

II. APPROACH

The goal of this project is to classify guitar chords from images using object detection techniques with neural networks. We have decomposed the entire task into four main sub-tasks: detection of the hand, fingers, frets and strings (see Fig. 1).

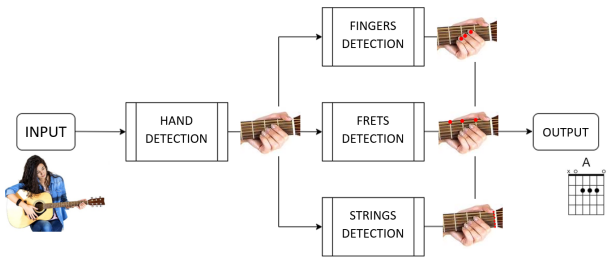


FIG. 1: Main parts of the chord detection approach.

In order to remove all redundant information, a YOLO network is used to detect the left hand of the guitar player. After cropping the image, it is sent through a Stacked Hourglass Network, where the fingers, frets and strings are detected. A modified 2-stack Hourglass network is used to perform the three tasks simultaneously (see Fig. 2). The structure of this network is similar to the one that Sanchez et al. used for multi-task human analysis in still images [5]. A stream of 2-stacked Hourglass Networks is used to learn each task. These streams are then integrated by adding intermediate connectivity and supervision. Given an input RGB image, a set of residual modules are applied in order to generate shared features among all network streams. Each module has

*Electronic address: a.mitjanscoma@gmail.com

an independent supervision (a different loss is applied to each one of them) and provide intermediate predictions (in the form of heatmaps), which are concatenated to form a tensor of size $64 \times 64 \times (\#stream \times 256)$, where 256 is the default number of Hourglass features. Next, each stream applies two different residual modules to the obtained tensor, before sending it through the second Hourglass module.

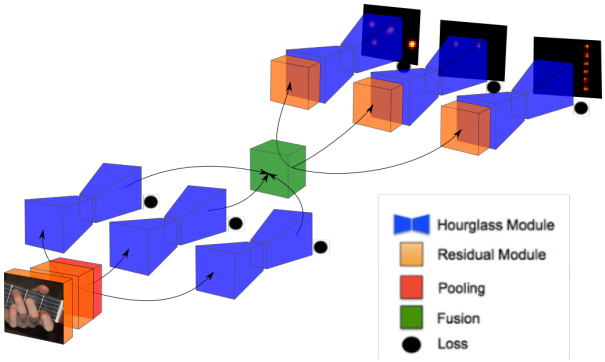


FIG. 2: Proposed multi task architecture.

This network outputs a heatmap for each task, where the network predicts the probability of the presence of a specific object at each and every pixel of the image.

To classify these obtained heatmaps as a specific chord, we designed an algorithm that not only deletes the false positives and adds the false negatives of the detection of the frets and the strings, but creates a tablature from the given outputs. This tablature is compared to the tablatures of several chords and a confidence value is obtained for each of these chords. Finally the tablature is classified as the chord with the highest confidence value.

A. Data

Thanks to the owners' permission, the following datasets were used to pre-train the networks:

- **Hand Dataset** [7] This dataset contains 13k copyrighted RGB images of different people with annotations of the position of every hand in the image.
- **SCUT-Ego-Gesture Dataset** [8] This dataset contains 59k copyrighted RGB images of hands with 16 different hand gestures. They are annotated with the position of its important features, such as the finger tips and joints.

To train the neural network on the overall problem of guitar chord detection (see Section II B), 275 free images were downloaded from Google: 95 images of people playing the guitar and 180 images of hands playing several chords. Each of the first set of images were labeled with the coordinates of the bounding box containing the left hand of the player. The remaining 180 images were labeled with the position of every visible finger, fret and string. The finger's coordinates were only annotated if they were pressing on any chord of the guitar. Therefore, the Hourglass network should be able to distinguish if a finger is pressing a string or not.

208 additional images were taken of three different people playing different chords (Fig. 3). These images were first annotated with the coordinates of the bounding box of the left hand, and afterwards with the coordinates of every visible finger, fret and string with respect to the previous labeled bounding box, using the same criteria as before. The ground truth annotations of the images in the dataset can be seen in Fig. 4.

FIG. 3: Images taken for finger, fret and string detection.

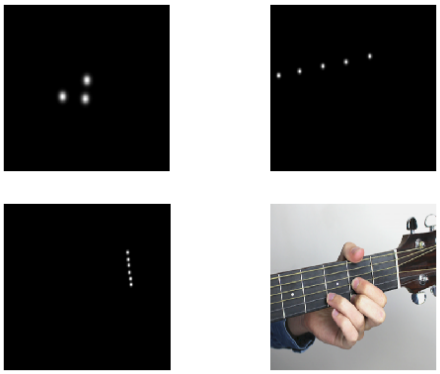


FIG. 4: Ground truth annotations of an image in the dataset.

B. Training

The YOLO network is pretrained with the hand dataset mentioned in Section II A, composed of 13k RGB images, and later trained with all the images containing a full representation of the body of the person (303 images). The Hourglass network is first pretrained with the SCUT-Ego-Gesture Dataset, which contains 59k RGB images. Since our task consists on detecting the finger tips of the left hand, only the first 11 gestures and the coordinates of the finger tips are used. Consequently, the network is pretrained with 41k images. Afterwards, the network is further trained with the 180 images obtained from Google and the 208 images taken. For data augmentation, transforms on images are randomly applied, including horizontal flips, random crops, random rotations and scaling. All YOLO input images are re-scaled into 416 $\times$ 416 pixels, and Hourglass input images are re-scaled and padded into 300 $\times$ 300 pixels.

Training in a GPU GeForce GTX 1080 Ti with 100 epochs takes about 158 minutes for YOLO and 720 minutes for the Hourglass network. A single forward pass takes about 0.1 seconds for YOLO and 0.3 seconds for the Hourglass network.

For supervision, a Mean-Squared Error (MSE) loss is applied for both networks, which measures the simi-

larity between the output and its corresponding target, as explained in Section V. In the Hourglass case, we perform a pixel-wise comparison between the predicted heatmaps and the ground-truth heatmaps, which consist of 2D gaussians centered on the objects locations. This loss is applied to the heatmap of every stacked hourglass of the model (in our case the network is formed by 2-stacked hourglasses). The final loss is just the sum of this two individual losses.

C. Chord classification

We designed an algorithm that deletes the false positives and adds the false negatives of the detection of frets and strings, by finding their extrapolated position based on the points detected by the network.

Afterwards, the tablature of the chord in the image is generated. This is done by projecting the points of every finger into the lines formed by the frets and the strings. By doing so, these projections can be compared to the values of the frets/strings to further establish the position of the finger.

To finally classify our image, the obtained tablature is compared to the tablatures of all the 14 major and minor chords. This comparison is made by setting a confidence value to each one of the 14 chords, which shows the similarity between the compared chords. The calculation of this value takes into account the position of the fingers and the total number of fingers used to play the chord.

III. RESULTS

In YOLO, evaluation is done by comparing the bounding box coordinates with the grounding truth. In the Hourglass network, it is done by comparing the maximums of the output's gaussians with the coordinates of all the visible frets, strings and fingers in the image. If a maximum falls within a certain distance from any of the objects that particular gaussian is considered as a true positive. The distance threshold has been set to 5 pixels. By computing the number of true positives, false positives and false negatives, the recall and precision of the network can be calculated with:

$$\text{Recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}} \tag{1}$$

$$\text{Precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}} \tag{2}$$

Basically Recall will show the percentage of features from a certain class that the network found with respect to the total number of objects of that class in the image. Precision is a calculation of the percentage of good answers of the network with respect to the total number of answers. It is important to always maintain a high precision, so that the network does not predict bad answers as well as a high recall, since it is required for the network to detect all the features in the image, if possible.

A. Hand detection

As said in Section I, a YOLO network is responsible for hand detection. When computing the network's performance on the 61 validation images, a 96% precision and 95% recall is obtained.

B. Fingers, frets and strings detection

For these tasks an Hourglass network is used, which provides 3 outputs in the form of a heatmap, containing gaussians in the locations of the detected objects. Two different approaches were taken before obtaining our final results.

1. Experiment 1

Firstly the Hourglass network was trained with the 388 images mentioned in Section II B. The performance of the three networks on the 80 validation images can be seen in Table I.

	Recall	Precision
Fingers	94%	83%
Frets	95%	98%
Strings	88%	97%

TABLE I: Results of the evaluation of the Hourglass network on the testing images.

2. Experiment 2

Due to the similarity between a finger tip with its nail hidden and the fingers' joints, it is very hard for the network to differentiate them. Consequently, it is logical to think that the network would perform much better if the nails were always visible. Therefore, the images mentioned in Section II A were retaken considering this hypothesis. As a consequence, the images downloaded from Google were discarded, since they did not comply with these requirements. When training the networks with these new 205 images, the following validation results were obtained.

	Recall	Precision
Fingers	93%	96%
Frets	97%	100%
Strings	99%	100%

TABLE II: Results of the evaluation of the Hourglass network on the testing images.

C. Chord classification

To measure the performance of the algorithm that extracts the tablature from the obtained heatmaps, the 205

newly taken images were used. These images contain a total of 14 different chords (major and minor chords) played in 17 different shapes (the G chord is played in three different ways and the C chord in two). Every image is passed through the YOLO network, where the left hand is detected. After cropping it, the image is sent to the Hourglass network, where the fingers, frets and strings are detected. After applying our algorithm, the tablature obtained is compared to every major and minor chord and then classified. This method performs with a 97% accuracy, classifying with success 199 out of 205 images.

C	24	0	0	0	0	0	0	0	0	0	1	0	0	0
Cm	0	12	0	0	0	0	0	0	0	0	0	0	0	0
D	0	0	12	0	0	0	0	0	0	0	0	0	0	0
Dm	0	0	1	11	0	0	0	0	0	0	0	0	0	0
E	0	0	0	0	12	0	0	0	0	0	0	0	0	0
Em	0	0	0	0	1	8	0	0	0	0	0	3	0	0
F	0	0	0	0	0	0	12	0	0	0	0	0	0	0
Fm	0	0	0	0	0	0	0	11	0	0	0	0	0	0
G	0	0	0	0	0	0	0	0	35	0	0	0	0	0
Gm	0	0	0	0	0	0	0	0	0	12	0	0	0	0
A	0	0	0	0	0	0	0	0	0	0	12	0	0	0
Am	0	0	0	0	0	0	0	0	0	0	0	12	0	0
B	0	0	0	0	0	0	0	0	0	0	0	0	13	0
Bm	0	0	0	0	0	0	0	0	0	0	0	0	0	12
	C	Cm	D	Dm	E	Em	F	Fm	G	Gm	A	Am	B	Bm

TABLE III: Results of the evaluation of the Hourglass network on the testing images.

Table III shows the confusion matrix for the results obtained. The detection is performed perfectly in almost all the chords except for the Em chord, which is confused for an Am or an E. This is because the network wrongly detects the index finger, which is not pressing any string (see Fig. 5). Consequently, the obtained confidence score is higher for the Am or E chords than for the Em chord.

This bad detection is probably due to the small amount of images in our dataset. By increasing the number of images containing an Em chord, the network should learn to detect its fingers properly.

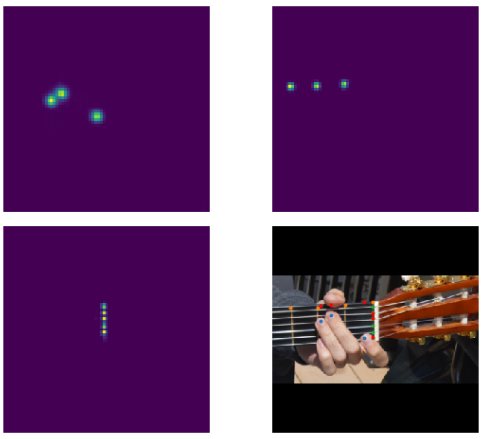


FIG. 5: Feature detection of an image of the Em chord.

D. Video classification

In order to fully discern this method's performance, 21 different videos were filmed of three different people playing 7 different songs each. Every frame of every video was classified and further reconstructed to create a video showing the detected chords with their confidence value (mentioned in Section II C). Given that the video contains 30fps, and it is unlikely that one chord is played for less than 0.2 seconds, the value of the detection is updated every 5 frames with the mean of the results obtained in these 5 frames, making this method resistant to sporadic errors. The overall result of this experiment was a successful detection of almost every chord.

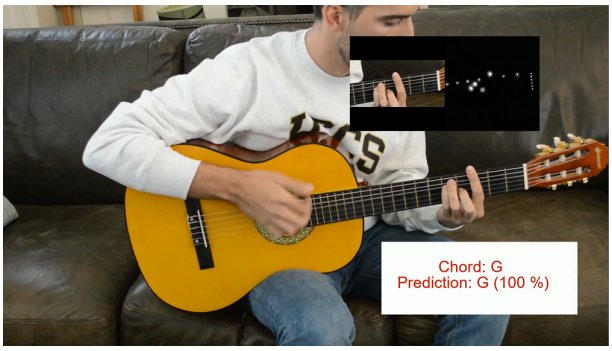


FIG. 6: Chord detection of a video frame.

IV. CONCLUSIONS

This paper presents a method based on machine learning techniques, using a feed-forward neural network, for guitar chord recognition, applicable to images. As no chords database seems to be publicly available, a dataset has been built containing images of different chords with three different approaches: 180 images containing chords played in different guitars and taken with different vision angles, 208 images of chords played in the same guitar but with different vision angles and finally 205 images of chords played in the same guitar with the constraint that all finger nails must be visible. Even though the first approach would make the detection of frets and strings more robust (since it does not have any constraint), the best results have been obtained using the third approach.

To the best of our knowledge, our paper presents the first method capable of detecting chords from images. Its high accuracy (97%) in the detection of 14 different chords makes it an exciting starting point for future improvements or applications.

For future work, both image chord detection and audio chord detection can be considered to make the detection much more robust. In addition, new chords can be added to the detection (C#, C7, ...), together with more finger shapes for each chord (played in higher frets).

Acknowledgments

I would like to thank both my advisors, professor Artur Carnicer in Barcelona and professor Jugal Kalita in Colorado, for their guidance and counsel. I would also like to thank Hector Pascual and Guillem Budia for letting me take pictures of them to build the dataset. Finally I want to express my infinite gratitude to the Balsells Mobility Program for funding this project.

[1] Choi, Keunwoo and Fazekas, György and Cho, Kyunghyun and Sandler, Mark, "A Tutorial on Deep Learning for Music Information Retrieval", *arXiv:1709.04396v2*, 2018.

[2] Ren, Shaoqing and He, Kaiming and Girshick, Ross and Sun, Jian, "Faster r-cnn: Towards real-time object detection with region proposal networks", *Advances in neural information processing systems*, pp. 91-99, 2015.

[3] Newell, Alejandro and Yang, Kaiyu and Deng, Jia, "Stacked Hourglass Networks for Human Pose Estimation", *European conference on computer vision*, pp. 483–499, 2016.

[4] Redmon, Joseph and Divvala, Santosh and Girshick, Ross and Farhadi, Ali, "You only look once: Unified, real-time object detection", *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

[5] Sánchez, Daniel and Oliu, Marc and Madadi, Meysam and Baró, Xavier and Escalera, Sergio, "Multi-task human analysis in still images: 2D/3D pose, depth map, and multi-part segmentation", *2019 14th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2019)*, pp. 1–8, 2019.

[6] Lihi Shiloh-Perl and Raja Giryes, *Introduction to deep learning*, (arXiv:2003.03253v1, Feb 20, 2020).

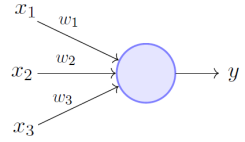
[7] Arpit Mittal, Andrew Zisserman and Phil Torr, *Hand Dataset*, `http://www.robots.ox.ac.uk/~vgg/data/hands/`

[8] Chenyang Li, *SCUT-Ego-Gesture Dataset* `http://www.hcii-lab.net/data/SCUTEgoGesture/index.htm`

V. APPENDIX

A. Deep Learning and Neural Networks

Deep Learning is a Machine Learning method based on deep artificial neural networks [6]. Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. These neural networks are composed of multiple layers: the input layer, several hidden layers and the output layer (see Fig. 8). Each layer is made of multiple nodes, also called neurons of Perceptrons.



$$y = f\left(\sum \omega_i x_i + b\right) \quad (3)$$

FIG. 7: Perceptron model.

A node (see Fig. 7) is a computational unit that has one or more input connections and an output connection.The output is calculated with Eq. 3, where ω_i are called the weights, x_i are the inputs, b is the bias and $f(x)$ the activation function, which commonly corresponds to the sigmoid function.

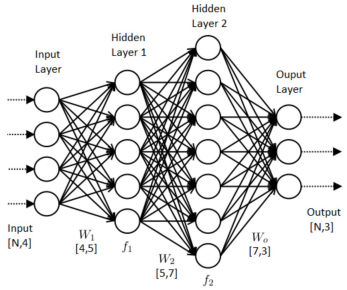


FIG. 8: Neural network model with two hidden layers.

Deep learning networks are distinguished from the single-hidden-layer neural networks by their depth (number of layers). More than three layers qualifies as "deep" learning. However, deep learning models can contain up to hundreds of layers and millions of nodes.

A key concept of neural networks is **Gradient Descent**. Gradient Descent is an optimization algorithm used to minimize some function moving in the direction of steepest descent defined by the negative value of the gradient. In deep learning gradient descent is used to update the parameters (weights and biases) of our model. First of all, a loss function is defined $L = \sum l_n$ (l_n being the loss of the output n), which compares every output of the network with the results we should have obtained (defined by the labels of the dataset). Then the gradient of each loss with respect to every parameter must be calculated (see Eq. 4). To do so, first the gradient of the loss with respect to the parameters of the output layer must be computed, and then **Back Propagation** is applied to compute the remaining gradient values.

$$\frac{\partial l_n}{\partial \omega_{i,j}^{(l)}} = \frac{\partial l_n}{\partial s_i^{(l)}} \frac{\partial s_i^{(l)}}{\partial \omega_{i,j}^{(l)}} = \frac{\partial l_n}{\partial s_i^{(l)}} x_j^{(l-1)} \; ; \; \frac{\partial l_n}{\partial b_i^{(l)}} = \frac{\partial l_n}{\partial s_i^{(l)}} \quad (4)$$

In Eq. 4, $s_i^{(l)}$ corresponds the value of the node i of layer l before applying the activation function. x is the value obtained after applying the activation function.

Finally, a Gradient Step is done by updating the parameters as in Eq. 5, where η corresponds to the **learning rate**, which defines the size of the update.

$$\omega_{i,j}^{(l)} = \omega_{i,j}^{(l)} - \eta \left(\frac{\partial l_n}{\partial \omega_{i,j}^{(l)}}\right) \; ; \; b_i^{(l)} = b_i^{(l)} - \eta \left(\frac{\partial l_n}{\partial b_i^{(l)}}\right) \quad (5)$$

This process is repeated until reaching the minimum of the loss function. The value of the learning rate must be set in order to reach the global minimum as fast as possible, but also to avoid falling into local minimums.