ECTI Transactions on Computer and Information Technology
Journal homepage: https://ph01.tci-thaijo.org/index.php/ecticit/
Published by the ECTI Association, Thailand, ISSN: 2286-9131

Advancing Guitar Chord Recognition: A Visual Method Based on Deep Convolutional Neural Networks and Deep Transfer Learning

Yosi Kristian[1], Lukman Zaman[2], Michael Tenoyo[3] and Andreas Jodhinata[4]

ABSTRACT

Initiated in 1999, Automatic Chord Recognition (ACR) primarily relied on audio data, facing challenges, especially with high timbre sounds, which led to a shift towards visual methods for recognizing guitar chords due to their distinct hand configurations. This project explores visual guitar chord identification, harnessing fretboard features, including hand arrangements and positions. It also investigates the limited advantages of transfer learning due to the absence of pertinent pre-trained weights. The developed model employs deep learning, DCNN methodologies, and techniques like normalization, flattening, and dropout to identify 14 major and minor keys without fret position restrictions. Enhanced by data augmentation, a self-compiled dataset of over 13,000 samples from 10 contributors effectively trains the model for new data. After testing 115 new examples, the system achieved an 83% accuracy on both live and pre-recorded video data. These results demonstrate the feasibility of employing a deep convolutional neural network (DCNN) focused visual approach for guitar chord identification. Furthermore, this study suggests exciting potential for future advancements in the Music Information Retrieval (MIR) field.

Article information:

Keywords: DCNN, Fretboard Detection, Transfer Learning, Chord Classification, Automatic Chord Recognition

Article history:
Received: November 1, 2023
Revised: February 22, 2024
Accepted: May 1, 2024
Published: May 4, 2024
(Online)

DOI: 10.37936/ecti-cit.2024182.254624

1. INTRODUCTION

Musical chords, crucial for music information retrieval (MIR), inform us about genre, playstyle, and tone. The Automatic Chord Recognition (ACR) field has continually evolved, as identified in [1], though challenges remain in optimizing chord detection. The development of ACR began in 1999 when a project utilizing lisp music [2] enabled chord recognition at the signal level. Consequently, many research works embarked on advancing the ACR system, predominantly focusing on audio data due to the underdeveloped state of visual programming and computer vision. However, some methods introduced, such as the EM-Trained Hidden Markov Model [3] in 2003, failed due to insufficient data generalization, evidenced by a low 40% accuracy.

Further advancements were made utilizing an aural method with chromagram [4], deep chromagram extractor [5], and spectrograms with recurrent neural networks[6] as researchers went back to Fujishima's 1999 approach[2]. Mel-Spectrogram with DCNN learning [7] optimized the unique features of human voices and musical instruments in audio data, outperforming earlier methods.

However, despite the successes and continued use of spectrogram-based features, improvements stagnated due to the inherent shortcomings of the aural approach in handling highly timbre sounds. The issue is intrinsic to the aural characteristic; hence, despite methodological developments such as multi-task learning with physical data augmentation[8], dissimilarity space for classification [9], and deep-learning variations [10], the same pitfalls persisted. Despite being widely employed, spectrograms' inability to handle high-timbre sounds has made musicians less confident in the method [1], limiting their use in applications such as online music education services [11].

Motivated by human perceptual abilities, where vi-

[1,2,3] The authors are the Informatics Department, Faculty of Science and Technology, Institut Sains dan Teknologi Terpadu Surabaya, Indonesia, E-mail: yosi@stts.edu, lz@stts.edu and michael32@mhs.stts.edu
[4] The author is the Informatics Department, School of Information Technology, Universitas Ciputra Surabaya, Indonesia, E-mail: andreas.jodhinata@ciputra.ac.id
[1] Corresponding author: yosi@stts.edu

sual recognition of chords often surpasses auditory recognition, this project proposes a shift to a visual ACR system. With the advancements in computer vision since The ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[12] and its increasing relevance in software development [13], this project explores the potential of using DCNN in visual ACR systems.

Few researchers examined visual ACR systems in the past. In 2005, experiments concerning the visual detection of guitar-finger movements were published by Anne-Marie Burns and Marcelo M. Wanderley [14]. The subsequent focus has been on using conventional image processing methods to extract hand patterns.

On marked fingers, subsequent research utilized Bayesian Classifier and Particle Filters [15], template matching [16], and conventional image processing techniques [17]. The final study to employ conventional image processing was conducted by Waseda University's Jun Ohya and Zhao Wang [18]. This project aims to continue this line of research and add deep learning techniques to improve visual ACR systems.

Regardless of the sophistication of the algorithms or methodologies used, all solutions employing image processing techniques share the same issues, including limitations on input data [19]. For instance, finger point detection and tracking struggle with the 'joined finger contours' problem, where an algorithm mistakenly detects multiple fingers as a single contour. As most guitar chords require stacked fingers, this is a significant challenge. Other complications include sub-optimal Hough line detection on blurry images and missing fingers. Here, deep learning plays a crucial role in addressing these limitations.

Two studies have incorporated deep learning as the core system, utilizing hand patterns as learning features. One used a standard DCNN with a pre-trained network [20], while [21] applied transfer learning.

This project aspires to develop a system capable of accepting an image of someone playing guitar, isolating the fretboard region, and feeding this region of interest into a DCNN model for chord prediction. The project aims to identify up to 14 fundamental guitar chords: C, D, E, F, G, A, B, Cm, Dm, Em, Fm, Gm, Am, and Bm. The final system, integrated into a website and accepting video data inputs in real-time or otherwise, offers a realistic testing environment.

This paper contributes to the field of automatic chord recognition (ACR) through the following means:

1) Our team created a fretboard detector that identifies the entire fretboard area. This detector serves as input for our chord classification system. As observed in previous studies, the key finding is that utilizing the full range of fretboard information leads to more accurate chord recognition than relying solely on hand pattern recognition. By considering the entire fretboard, we enhance the precision of chord identification. [20][21]. This advancement helps overcome limitations in chord class identification.
2) This research investigates the effectiveness of transfer learning in constructing our classification model, considering the absence of pre-established weights in this domain. By illustrating that transfer learning does not invariably outperform a model constructed from the ground up, we hope to illuminate circumstances in which transfer learning should be implemented.
3) Instead of merely adapting existing architectures, we modify the final layer, incorporating our unique architecture with superior performance. We expect our model to serve as a fundamental framework for subsequent investigations into the classification of visual guitar chords.
4) We assess the system's performance using a realistic testing environment that handles real-time and pre-recorded video data. We intend to showcase the significant potential this approach holds for future advancements in the field of ACR.

2. RELATED WORKS

Over recent decades, computer vision has evolved into a rapidly expanding and influential field of study[13]. It has found applications across various domains, including autonomous vehicles [22], traffic violation detection systems [23], and automatic identification in the industrial sector. The catalyst for this explosion of interest and development was the ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [12], announced in 2012, which ignited a series of advancements in deep learning applied to computer vision. Despite these developments, the Automatic Chord Recognition (ACR) system has witnessed a comparatively slower adoption of the visual approach. It was not until 2018 that computer vision and deep learning integration were applied to ACR, as evidenced in two significant papers [20] [21].

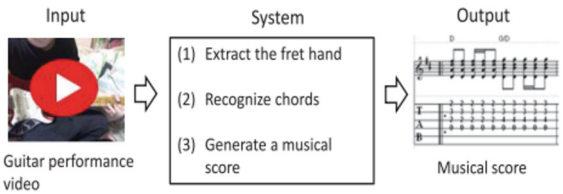

Fig.1: *System Configuration of Previous Related Work.*

Both proposed systems exhibit similar workflows, as illustrated in Figure 1. Each of them utilizes hand pattern information to identify guitar chords. They detect the hand within the image and then send the detected region to a convolutional network for chord

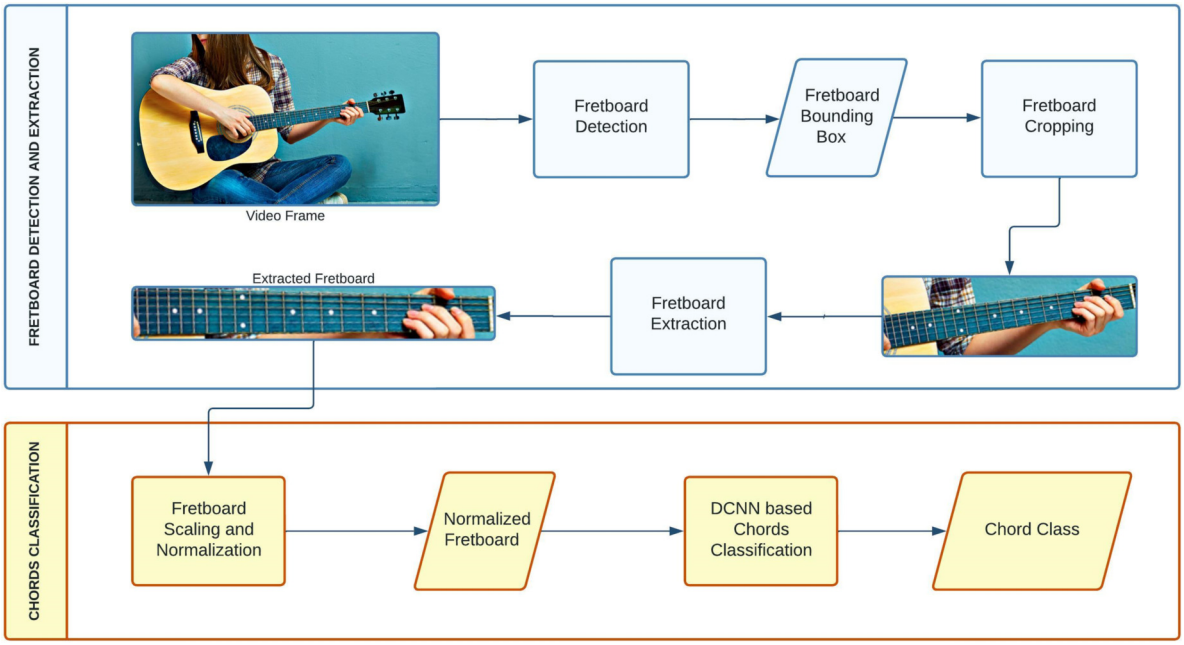

Fig.2: *Our Main System Architecture.*

classification. Despite using the same LeNet (Inception) methodology, they yield distinct results.

The preliminary system described in [20] attempts to identify the cropped finger pattern by automatically generating a corresponding musical score using VaticJS. Three distinct scenarios, assigned to class clusters of three, five, and six, were utilized to evaluate the system's effectiveness. It documented a recognition rate of 55% for a set of six chords. (A, B, C, D, E, F, G). Significantly, the system's performance improved when the class size was decreased: a three-chord classifier demonstrated a recognition rate of 90%, while a five-chord classifier achieved a recognition rate of 70%.

The second study presented in [21] employs a distinct architecture for hand detection despite employing a comparable methodology. The hand region is cropped by the system using deep-learning object detection, which divides the process into hand detection and chord classification. The hand detection model underwent training using the EgoHands dataset on the MobileNet v2 network. For comparison purposes, the classification model was evaluated on numerous networks. Using pre-trained LeNet from ImageNet weights, the project achieved a perfect accuracy of one hundred percent, according to the report; nevertheless, it performed inadequately on new testing data. Both studies have identified LeNet as the most effective network for this classification task, indicating that it may serve as a beneficial foundation for our research or even be enhanced through our adjustments.

Timothius Tirtawan *et al.* [24] proposed a technique for semantic segmentation using a Conditional Generative Adversarial Network (CGAN) with U-Net architecture to apply batik patterns to clothing in images. This study highlights the potential of Convolutional Generative Adversarial Networks (CGANs) in the fashion industry. It contributes to the broader goal of implementing deep learning techniques in the visual arts, which is pertinent to visual guitar chord recognition.

The study by Widjojo et al.[25] presents an innovative approach to expedite car insurance claims in Indonesia through an integrated deep learning system. The system employs deep transfer learning to segment automobile damage, identify damaged components, and categorize the extent of the damage. The segmentation tasks were performed using Mask R-CNN, while the classification tasks involved a comparison of EfficientNet and MobileNetV2. To enhance the F1 score, they implemented a straightforward CNN concatenation. Notably, the model that exhibited the highest performance was MobileNetV2 with CNN modifications, achieving an impressive F1 score of 91%. This approach is consistent with our visual method for identifying guitar chords, in which deep transfer learning is crucial to attaining high precision and productivity.

Building upon the foundation laid by previous research, our work aims to address the limitations of existing ACR methods by leveraging the full range of fretboard information and investigating the efficacy of transfer learning in training the classification model. By proposing a unique architecture and rigorously assessing the system's performance in a realistic testing environment, we contribute to the advancement of visual ACR systems and their potential for future developments in the field of MIR. Our approach pushes the boundaries of chord recognition accuracy and opens

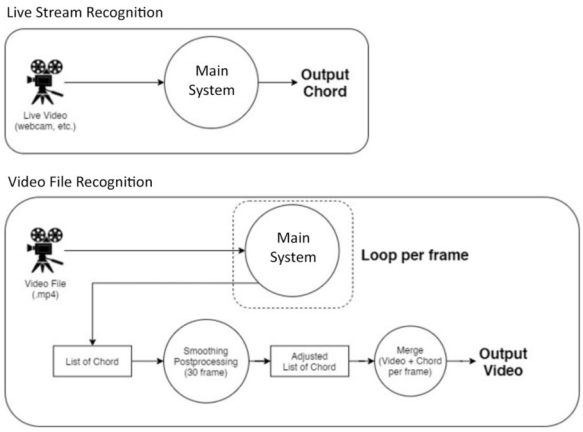

Fig.3: *Web Features Architecture - Upper Flow for Live Recordings, Lower Flow for Video Files.*

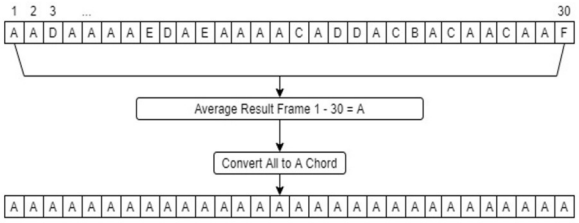

Fig.4: *Example illustration of mean mode post-processing with leap length of 30.*

up new possibilities for integrating ACR systems in various applications, from music education to interactive music experiences. Through this work, we hope to inspire further research and innovation in the exciting intersection of computer vision, deep learning, and music information retrieval.

3. PROPOSED WORKS

This research project's architecture comprises several interconnected subsystems, as depicted in Figure 2. It is crucial to note that Figure 2 exclusively illustrates the workflow for chord classification and omits the complete system architecture encompassing video and live-stream data processing.

3.1 Fretboard Detection

In contrast to previous research that primarily utilized hand pattern recognition to identify guitar chords, this research employs the entire fretboard to classify chords. The reason is that despite similar hand patterns across the fretboard, numerous chords differ in their placements, complicating their categorization based solely on hand patterns. The initial pipeline, divided into two sections, will be tasked with extracting the fretboard or neck of the guitar.

As illustrated in Figure 2, the fretboard detection subsystem (highlighted in blue) employs a Single Shot Detection (SSD) model undergirded by a MobileNetV2 base model, pre-trained on the EgoHand dataset. The subsystem processes the input image or video frame using a Deep Convolutional Neural Network (DCNN) model. The model generates coordinates for bounding boxes that outline the fretboard, the Region of Interest (ROI). The image is cropped with an additional 15% tolerance on y-max (top) and x-max coordinates relative to the ROI (right). Additional preprocessing is then applied to this cropped image before it is passed through the subsequent pipeline.

3.2 Chord Classification

This subsystem extends the fretboard detection pipeline to classify fourteen unique guitar chords. The model's customized architecture, partially based on the InceptionResnetV2 framework and supplemented with extra layers, is illustrated in Figure 2 (in the yellow region). A series of experiments were conducted, including the implementation of transfer learning. As a result, we developed an optimized model that has achieved significant performance metrics.

3.3 Realistic Testing Environment

Our research aims to go beyond theoretical contributions and demonstrate the chord recognition system in a practical testing environment. The video data-processing system is integrated into a web-based interface that uses the Django framework.

Two operational scenarios are delineated in Figure 3: the analysis of pre-recorded video data and real-time chord recognition. The previous system analyzes real-time feeds from webcams or other recording devices and classifies and instantly presents the chord type. It adheres to the fundamental workflow, which consists of classifying chords, extracting fretboard information, preprocessing, and outputting the final chords, emphasizing speed and precision.

The analysis of pre-recorded videos that have been meticulously curated pertains to the latter scenario. Although the fundamental architecture remains consistent with the real-time model, additional postprocessing stages are implemented to mitigate noise and improve the output quality. The end product is a fusion of the initial video data and chords detected frame by frame.

3.4 Image Preprocessing

This research project employs various image preprocessing techniques to optimize the performance of the Deep Convolutional Neural Network (DCNN) model. The objective is to identify the most effective preprocessing strategy for enhancing model accuracy. Several preprocessing techniques were evaluated; however, the "fretboard flatten" method was deemed unsuitable for the dataset because of inconsistencies. The techniques under evaluation include

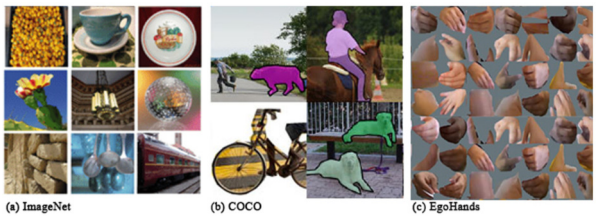

Fig.5: *Pre-trained Datasets. Consists of 3 types: (a) Example of ImageNet, (b) Example of COCO, (c) Example of EgoHands.*

Fig.6: *Example of Fretboard Datasets: The image shown above is a raw, unedited example that includes annotations made using LabelIMG.*

Min-Max normalization within the range of -1 to 1, grayscale input conversion, and histogram equalization in both RGB and grayscale formats. A comparative analysis was undertaken to determine which preprocessing strategies yielded the highest level of performance for the model.

3.5 Output Postprocessing

Anomalies such as misclassifications may arise due to the fully automated nature of the chord recognition process, which could result in inconsistent and potentially disruptive output for end-users. For example, the user may experience confusion in response to rapid chord changes within a brief frame sequence. A postprocessing mechanism has been implemented in response to these concerns to reduce the frequency of irrational chord changes and stabilize the output. The objective of this stage is to better synchronize the system's output with the cognitive abilities of a guitarist, who is physically incapable of performing multiple chord changes in a single second.

The "Mean Mode" postprocessing algorithm is implemented to guarantee the chord recognition output's consistency. By employing this methodology, the system determines the chord (mode) that occurs most frequently within an array of a predetermined length and modifies every entry to correspond with that mode. A visual representation of this algorithm can be found in Figure 4. The array length associated with this mechanism is configured to align with the frame rate of the video. As an illustration, when the video is being processed at 30 frames per second, the Mean Mode algorithm will require an array of 30 frames in length.

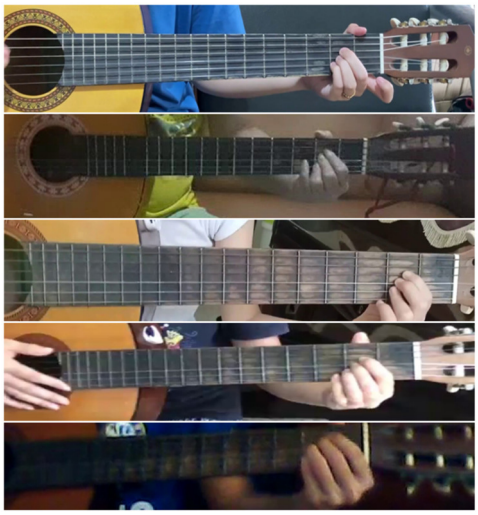

Fig.7: *Example of Chord Datasets.*

3.6 System Limitations and Constraints

The current implementation of this project is subject to several limitations and operational constraints, detailed as follows:

1) Offline Operation: The system is designed to function in an offline environment.
2) Camera Stability: Minimal to no camera movement is required for optimal performance.
3) Input Specifications: At least one guitar must be in the input image. Failure to meet this condition can result in a critical error.
4) Video Criteria:
   a. The camera angle must be front-oriented.
   b. The guitar neck should be entirely visible, without any partial cropping.
   c. The fretboard lines must be discernible and not overly dark.
   d. The camera angle ought not to be tilted excessively.
   e. The input quality should be low-noise and sufficiently sharp, clearly displaying the guitar neck.
5) Accuracy Variability: System performance may differ based on the specific input scenario.
6) The system's chord classification range is limited to predefined chords.
7) Pattern Sensitivity: There is a high likelihood of misdetection if unconventional or cluttered hand patterns are used.
8) Handedness: The system is not compatible with left-handed guitar players.
9) Guitar Specifications: Only standard, capo-free guitars are compatible.

10) File Format: The system supports only video files with extensions .mp4, .wmv, and .avi.

4. DATASETS

This study uses three types of datasets: Pre-trained Datasets, Fretboard Datasets, and Chord Datasets. Since no publicly available datasets exist for this topic, we created the last two datasets ourselves. Pre-trained Datasets are crucial in evaluating whether transfer learning can enhance our model's performance.

4.1 Pre-trained Datasets

We use the MobileNet V2 and MobileNet V1 models, which were trained on the COCO and EgoHand datasets, to build our fretboard detection model. We used ResNetV3 and InceptionResNetV2 models trained on the ImageNet dataset to identify chords. While transfer learning is not mandatory, we selectively employ it only when it demonstrably improves our model's performance. Should empirical testing reveal its ineffectiveness, we remain open to its exclusion. Our interest in this field stems from successfully utilizing transfer learning methods in prior research [21].

As shown in Figure 5, this study uses three different pre-trained datasets to start off our neural networks. Each dataset has its features and is used for a specific task in the project, like guitar chord classification or fretboard detection.

1) ImageNet Dataset: This Dataset is used only for classifying guitar chords. It has over 14 million images that cover 20,000 types of natural objects like "balloons," "dogs," and "cats" [12]. It is an extensive assemblage comprising a wide variety of natural entities.
2) The COCO dataset, "Common Objects in Context," is employed to identify fretboards. The dataset is of considerable size and is dedicated to object identification. Approximately 200,000 labeled images are organized into 80 distinct categories [26]. Although somewhat comparable to ImageNet, the COCO dataset possesses a distinct emphasis.
3) The EgoHands Dataset is utilized to assess and compare the performance of fretboard detection algorithms. The dataset includes over 15,000 hand images with high-quality labels[27].

4.2 Fretboards dataset

The images in this second dataset category relate to guitar fretboards and are utilized primarily to train detection models. Given the unavailability of publicly accessible fretboard datasets, we curated a collection of 115 images by sourcing them through Google search. This dataset was subsequently partitioned into testing and training subsets. Specifically,

Table 1: *Data Contribution for Chord Classification.*

Chord Class	Number of Images
C	998
D	1,025
E	1,004
F	995
G	985
A	1,002
B	974
CM	983
DM	859
EM	849
FM	900
GM	929
AM	928
BM	1,026
Total	**13,457**

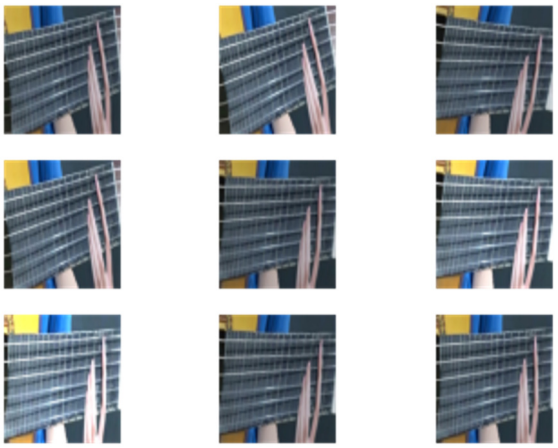

Fig.8: *Data Augmentation Examples.*

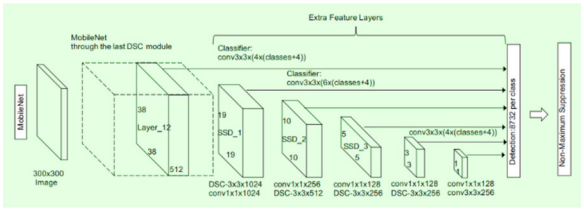

Fig.9: *Single Shot Detection (SSD) Model Combined with Mobilenet As the Base Model for Our Feature Extraction.*

the testing group comprises 22 images (approximately 20 percent), while the remaining 93 images form the training set (approximately 80 percent). Noteworthy instances from this dataset are visually depicted in Figure 6.

LabelIMG Tools is utilized to annotate the fretboard regions once the images have been gathered. The training and testing sets are subsequently optimized for utilization with the Detection Module of TensorFlow. [28]. We store every image-related detail in a CSV file, including their labels and location.

For ease of use, this file is subsequently converted to TFRecord format. There is only one image class in this dataset: "fretboard."

4.3 Chords Dataset

The Chord Datasets, similar to the Fretboard Datasets, comprise visual representations of guitar fretboards. In contrast, these illustrations illustrate particularized hand positions corresponding to distinct guitar chords. Due to the scarcity of these particular images on the internet, we needed to compile our dataset.

We recorded videos of 10 people playing 14 unique guitar chords to do this. Each person spent about 3 seconds on each chord. We then took frames from these videos and turned them into RGB images to make up this dataset. We used our fretboard detection system, built in the first part of the project, to automatically find and focus on the fretboard areas in these images. Examples of images, especially those showing the A Major chord, can be seen in Figure 7. The examples shown above are the original, unedited images presented this way to make it easier for the reader to understand what they look like before the fretboard is isolated. In total, the Chord Datasets have 13,457 images. Table 1 shows how many images we have for each chord, giving a complete picture of what's in the dataset.

Table 1 shows how many images are in each chord class in the final dataset. While compiling the dataset, we found and removed some images where the fretboard wasn't correctly captured. These issues were spotted through manual checks.

As the research advanced, dataset modifications became necessary due to a lack of data diversity and constraints in our computer systems, like memory and hardware capabilities. Data augmentation techniques enriched the dataset, altering image brightness and applying random rotations to introduce variety. Figure 8 illustrates examples from chord data post-augmentation. Following augmentation, standard preprocessing was performed on the images, preparing them for machine-learning tasks.

5. MACHINE LEARNING MODELS

Several robust DCNN architectures have been made publicly available through the ILSVRC competition [29]. These models, developed by leading researchers, have demonstrated solid performance, especially in the competition. Instead of investing time in constructing a DCNN architecture from scratch, beginning with one of these established designs is practical. This strategy enables us to concentrate on adapting an existing model to meet the unique requirements of this project.

5.1 Single Shot Detection (SSD)

SSD provides a time-saving approach to object detection. Unlike other techniques, SSD skips the separate object proposal processing and carries out all tasks within a single neural network. Tests on PASCAL VOC, COCO, and ILSVRC datasets reveal that SSD is faster and highly accurate [30]. For example, it reached a 74.3% mean Average Precision (mAP) on the VOC2007 test while operating at 59 FPS on an NVIDIA TitanX. With a 512 × 512 image input on the same Dataset, SSD surpassed Faster R-CNN, achieving a 76.9% mAP. It also demonstrated better accuracy compared to other 'single stage' methods.

SSD uses a feed-forward Deep Convolutional Neural Network (DCNN) to produce bounding boxes and their confidence scores. It then applies non-maximum suppression to finalize the detection results. SSD is typically built on top of a base network, such as MobileNet V1 or MobileNet V2, which helps generate feature maps from the input images. Unlike other models, SSD relies on this base network for feature extraction and cannot operate independently.

5.2 MobileNetV1

MobileNet is a Deep Convolutional Neural Network (DCNN) architecture created by Andrew G. Howard at Google in 2017[31]. The model aims to make DCNNs faster and more compact, making them suitable for devices with limited memory, like mobile phones or embedded systems. MobileNetV1 uses depth-wise Separable Convolution operations to speed up processing. Tests on the TensorFlow Model Zoo show that the MobileNetV1 SSD model performs well (21 mAP) and is faster (30 ms) on the COCO dataset than other models.

5.3 MobileNetV2

MobileNet version 2, introduced in April 2017[32], builds upon the original V1 by incorporating two new features: Linear Bottleneck and Shortcut Connection between Bottlenecks. These additions help convert lower-level inputs into higher-level descriptors and speed up the model's training process. Compared to its predecessor, MobileNetV2 is more efficient, using up to twice fewer operations, 30% fewer parameters, and running 30%–40% faster on a Google Pixel Phone while maintaining or even improving accuracy. This result shows it is highly effective for object detection and segmentation tasks. For instance, when tested on the COCO object detection task, both versions achieved the same performance (22.1 mAP), but MobileNetV2 was faster (200 ms vs. 270 ms). Overall, the second version is 20 times more efficient and 10 times smaller while keeping performance levels the same.

5.4 Xception

Xception is a type of neural network that uses a specialized layer known as depthwise separable convolution. However, it makes a few changes to the original design [33]. First, it changes the order of operations: it does the 1x1 convolution before the channel-wise convolution. Second, it doesn't use intermediate ReLU activation, which is commonly found in other architectures. Tests show that staying with ReLU activation makes Xception more accurate than versions that use either ELU or ReLU.

5.5 InceptionV3

It is instructive to examine InceptionV3 development history to understand the model better. The initial version, known as GoogleLeNet or Inception-v1, was created by Christian Szegedy and colleagues [34]. In this version, the input is processed through multiple convolutional layers with varying filter sizes—1x1, 3x3, and 5x5—and undergoes max pooling. These different outputs are then combined. This architecture eliminates the need to select filter sizes for each layer manually.

The subsequent version, frequently termed BN-Inception, was also developed by the same team in 2015 [35] [36]. The architecture remains similar to the first version, with inputs processed through convolutional layers of 1x1, 3x3, and 5x5 sizes and max pooling. These are then concatenated to produce the output. The notable addition in this version is batch normalization, as indicated by the "B.N." prefix. Batch normalization facilitates more efficient training by mitigating internal covariate shifts. The Rectified Linear Unit (ReLU) is used as the activation function to tackle issues related to saturation and vanishing gradients.

In the third iteration, factorization techniques are introduced to reduce dimensionality, decreasing the risk of overfitting. This version also adds a grid size reduction module to reduce computational costs while maintaining network efficiency.

5.6 InceptionResnetV2

This model combines features from the top two architectures that won the ILSVRC2015 competition: Resnet and Inception-v3. It was developed by the same team that introduced the Inception model [37]. This model may be perceived as a sophisticated iteration of the initial Inception framework. As described in the accompanying paper, the model incorporates three Inception modules and includes shortcut connections on the left side of each module. In terms of computational cost, InceptionResnetV2 is comparable to Inception-v4. However, it offers advantages in training speed and achieves slightly higher final accuracy. The primary distinction between the two InceptionResnet versions lies in the Inception module used; the second version employs the module from Inception-v4.

Table 2: *Detection Model Results.*

Version	Base Model	Initial Weight	LOSS
v1	SSD MobileNet v1	COCO	0.8943
v2	SSD MobileNet v2	COCO	0.7543
v3	SSD MobileNet v2	EgoHands	0.7697
v4	**SSD MobileNet v2**	**EgoHands**	**0.6616**
v5	SSD MobileNet v2	None	0.7205

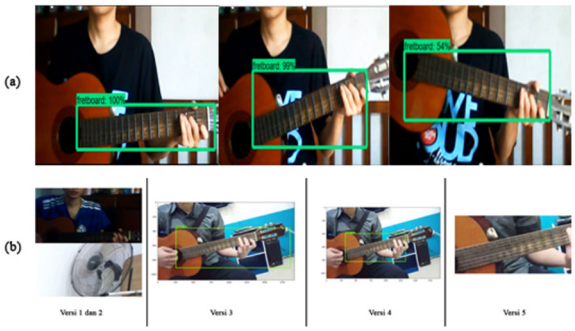

Fig.10: *Results of Fretboard Detection Model in Real-time Webcam Input, and Chord Dataset. (a) Results on frames from real-time testing. (b) Results on samples from our chord dataset.*

5.7 Addon Layers and Optimizers for Classification Models

Based on empirical observations, enhancing the existing architecture with additional layers, often termed the "tail network," has effectively improved the accuracy of the DCNN model [38]. These supplementary layers are assembled in a trial-and-error manner, akin to solving a puzzle, until the most effective configuration is identified. The tail network is appended to the end of the existing architecture after removing the last layer from the original network. They are then tailored to fit the specific classes relevant to this project.

Various techniques are employed to construct this tail network, including flattening, average pooling, dense layer, and dropout. Flatten layers convert the multidimensional input data into a one-dimensional vector fed into the dense layers [39]. Average pooling layers reduce the spatial dimensions of the input while retaining essential features [40]. Dense layers are fully connected layers that perform the final classification task [41]. Dropout layers prevent overfitting by randomly setting a fraction of input units to zero during training [42].

Multiple optimizers such as Adam, SGD, and LAMB have also been experimented with to evaluate their impact on the model's performance. Adam is an adaptive learning rate optimization algorithm that computes individual learning rates for different parameters [43]. SGD is a simple yet effective optimization algorithm that updates the parameters in

Table 3: *Preprocessing Variation Trial Results.*

Preprocess	Train Acc.	Val. Acc.	Testing Acc.
Sample Wise Center + /255	1	0.989	0.52
Sample Wise Center	**0.994**	**0.996**	**0.67**
Grayscale Hist. Eq.	0.2	0.09	NaN
RGB Hist. Eq.	0.978	0.976	0.24

Table 4: *Base Model Variation Trial Results.*

Base Model	Train Acc.	Val. Acc.	Testing Acc.
InceptionV3	0.994	0.996	0.67
Xception	0.07	0.06	NaN
MobileNet V2	0.071	0.06	NaN
InceptionResnet v2	**0.999**	**0.983**	**0.702**

the opposite direction of the gradient [44]. LAMB is a layer-wise adaptive batch optimization technique that helps train deep neural networks with large batch sizes [45].

6. MODEL DEVELOPMENT

The development of the guitar chord recognition system involved two main components: the fretboard detection model and the chord classification model. This section describes the iterative process and experiments conducted to develop these models.

6.1 Fretboard Detection Model Development

The fretboard detection model aims to identify the fretboard area in guitar images accurately. Five iterations of the model were developed through a systematic process of trial and error. Each version represents a stage in the model's development and refinement. Some iterations were explicitly created for comparative analysis, and instances where the model failed to produce meaningful results were excluded from the record.

The base model used for all versions was SSD MobileNet, with variations in the version (v1 or v2) and initial weights (COCO or EgoHands). The evaluation criteria for model selection included the loss value, performance in real-time testing using a webcam, and the misidentification rate on the chord dataset.

Version 1 (v1) used SSD MobileNet v1 initialized with COCO weights. In versions 2 and 3, SSD MobileNet v2 was utilized, with initial weights of COCO and EgoHands, respectively. In addition, Version 4 (v4) utilized SSD MobileNet v2 with initial weights provided by EgoHands, which were subsequently refined. SSD MobileNet v2 was implemented in Version 5 (v5) without initial weights.

The models were trained to minimize the loss value, aiming for a loss under 1.0. Real-time webcam testing was performed to assess the models' hand-detection performance. However, the primary benchmark for model selection was the chord dataset, as the primary objective was to extract the fretboard area from this dataset accurately.

Table 5: *Resolution and Data Augmentation Variation Trial Results.*

Resolution	Data Augmentation	Train Acc.	Val. Acc.	Testing Acc.
114	No	0.999	0.980	0.648
114	Yes	0.999	0.980	0.702
229	No	0.992	0.991	0.730
229	**Yes**	**0.984**	**0.992**	**0.832**

Table 6: *Transfer Learning Variation Trial Results.*

Base Model	Unfreezed Layer After (x)	Testing Acc.
InceptionV3	mixed8	0.325
InceptionResNet V2	mixed_7a	0.643
InceptionResNet V2	**NO TL**	**0.832**

Table 7: *Optimizers Variation Trial Results.*

Optimizer	Learning Rate	Train Acc.	Val. Acc.	Testing Acc.
Adam	**0.001**	**0.984**	**0.992**	**0.832**
LAMB	0.001	0.995	0.937	0.665
SGD	0.001	0.995	0.944	0.656
Adam	0.01	0.07	0.05	NaN
Adam	0.0001	0.997	0.948	0.641

6.2 Chord Classification Model Development

Developing the chord classification model posed significant challenges and required a systematic approach. The process involved selecting preprocessing methods, base models, input resolution, data augmentation techniques, transfer learning, and optimizers.

Initially, InceptionV3 was chosen as the base model due to its proven effectiveness in previous research. However, the first attempt failed due to issues with fretboard normalization in the preprocessing stage. Subsequent trials employed different preprocessing variations, including sample-wise centering, normalization, grayscale conversion, and histogram equalization.

Several base models were evaluated, including InceptionV3, Xception, MobileNet V2, and InceptionResnetV2. The models were assessed using a 144x144 input resolution and data augmentation methods. InceptionResnetV2 emerged as the most efficient base model, showcasing near-perfect performance on the training and validation datasets.

As the research advanced, dataset modifications became necessary due to a lack of data diversity and constraints in our computer systems, like memory and hardware capabilities. Data augmentation techniques enriched the dataset, altering image brightness and applying random rotations to introduce variety. Data augmentation is a widely used technique in deep

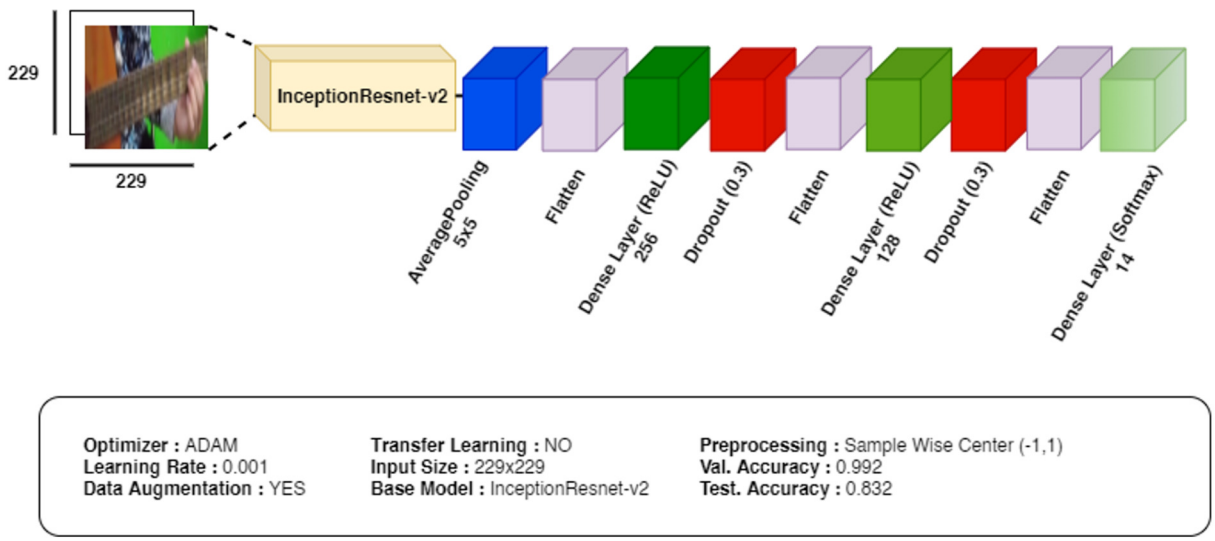

Fig.11: *The Final Version of DCNN Model Used for Chord Classification Task.*

learning to artificially increase the size and diversity of the training dataset [46]. It helps reduce overfitting and improves the model's generalization ability.

We applied brightness adjustment and random rotations as data augmentation techniques. Brightness adjustment involves increasing or decreasing the pixel intensities of the image by a random factor [47]. This augmentation helps the model learn to recognize chords under different lighting conditions. Random rotations involve rotating the image by a random angle within a specified range [48]. Augmentation also helps the model recognize chords from different angles and orientations. Figure 8 illustrates examples from chord data post-augmentation.

Finally, different optimizers were evaluated, including Adam, SGD, and LAMB. With a learning rate of 0.001, the Adam optimizer achieved the highest performance for the current model. The final model architecture incorporates the InceptionResnetV2 base model, sample-wise normalization preprocessing, a 229x229 input resolution, data augmentation techniques, and the Adam optimizer with a learning rate of 0.001.

7. RESULTS AND DISCUSSION

This section presents a comprehensive evaluation of the proposed model, assessing its effectiveness and performance in the domain of guitar chord recognition. The experiments were designed to rigorously test the model on various fronts, including accuracy, computational efficiency, and generalizability.

7.1 Fretboard Detection Results

The fretboard detection model versions were evaluated based on their loss values, performance in real-time testing, and misidentification rate on the chord dataset. Table 2 summarizes the loss values achieved by each model version.

Real-time testing using a webcam demonstrated that all models, except the last version (v5), exhibited good performance in hand detection. However, real-time testing alone was not decisive for model selection, as the chord dataset served as the primary benchmark.

Figure 10 illustrates instances of misidentification by each model on the chord dataset. The first and second versions (v1 and v2) occasionally misidentified common objects like clothing or people as the fretboard. The third version (v3) performed well unless the fretboard was tilted, leading to detection errors. The fourth model (v4) excelled in accurate fretboard detection across the chord dataset and was chosen as the final fretboard detection model. While precise in fretboard detection, the last version (v5) missed the hand, mirroring its performance in real-time testing.

7.2 Chord Classification Results

The chord classification model underwent rigorous evaluation using several datasets: 80% of the chord dataset for training, 20% for validation, and an additional 115 new images for testing to avoid overfitting. Models that performed poorly on the training and validation datasets did not progress to the testing phase.

Table 3 summarizes the results of different preprocessing variations. Sample-wise centering without division by 255 yielded the best testing accuracy of 0.67, albeit with some overfitting. This preprocessing method was chosen for further model development. Table 4 presents the performance of different base models. InceptionResnetV2 was the most efficient base model, displaying near-perfect performance on the training (0.999) and validation (0.983) datasets. Although it did not meet the predefined accuracy threshold above 0.8 on the testing dataset,

it

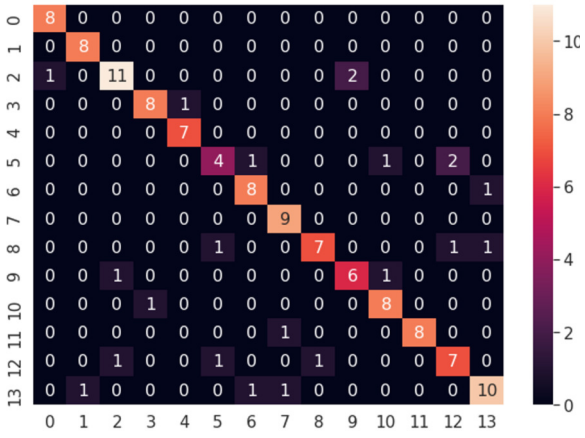

Fig.12: *The confussion matrix of the final model peformed on the testing dataset.*

showcased the best performance among the evaluated models and was selected for further development.

Experiments were conducted to examine the impact of input resolution and data augmentation on model performance, as shown in Table 5. Higher input resolution and data augmentation techniques were beneficial in enhancing model accuracy. The most accurate outcomes were attained in the final setup, incorporating both data augmentation and a higher input resolution of 229x229.

Transfer learning approaches were explored but did not significantly improve model accuracy compared to the model developed without transfer learning, as shown in Table 6. Table 7 presents the results of experiments with different optimizers. With a learning rate of 0.001, the Adam optimizer achieved the highest performance for the current model.

The finalized model architecture, showcased in Figure 11, incorporates the InceptionResnetV2 base model, sample-wise normalization preprocessing, a 229x229 input resolution, data augmentation techniques, and the Adam optimizer with a learning rate of 0.001. Figure 12 presents a confusion matrix to assess the model's performance. The model's overall accuracy is 83.21%, consistent across diverse classes.

7.3 Main System Experiments and Results

The system was tested in a realistic environment using video data via a web-based platform to validate the model's practical applicability. The platform featured real-time chord detection and pre-recorded video processing functionalities. The real-time feature was initially prioritized, with the system's output closely examined to ensure it included all necessary information, such as the raw input image, fretboard bounding box, FPS data, most probable guitar chord prediction, and confidence levels for each chord class. Preliminary real-time testing confirmed satisfactory performance, albeit with some limitations.

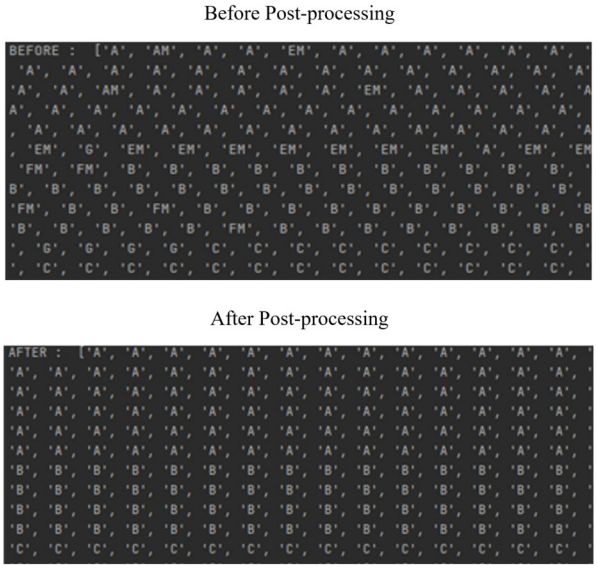

Fig.13: *Continuous Line Model Creation in Teachable Machine Program.*

The chord detection aspect performed well, accurately identifying most chords. However, misidentifications occurred due to similar hand patterns (e.g., A Minor and E), lighting conditions affecting fretboard area detection, excessive guitar slope, unconventional chord patterns, and inaccurate finger positioning. Specific chords like B Major and those with similar hand patterns (e.g., D and Dm, Am and E) posed detection challenges. The system's speed assessment revealed an average frame rate of 3 FPS, sufficient for human perception considering the resource-constrained development environment, but further enhancements are desired. The model surpassed its predecessor in loss and accuracy metrics and demonstrated effectiveness in realistic testing scenarios.

Pre-recorded video data and testing yielded consistent outcomes and limitations compared to real-time testing, with the system operating at an average speed of 3 FPS. For a 27-frame video, the estimated processing time would be 9 seconds. However, this test revealed another challenge: the robustness of chord changes. Unlike real-time processing, pre-recorded video allows additional computational steps without time constraints, enabling more comprehensive output postprocessing. Further details on this postprocessing phase will be discussed subsequently.

7.4 Effect of Postprocessing

A postprocessing technique called 'mean mode' was employed to refine the system's output by identifying the most frequently occurring output over specific time intervals. Figure 13 illustrates the impact of this technique on one of the test datasets during the debugging phase.

Before postprocessing, the output contained significant noise, with occasional misidentifications. The

postprocessing step mitigated these inaccuracies, resulting in a more coherent output. For instance, in the initial segment of the chord array, which is primarily class A, the postprocessing eliminated occasional misidentifications of other classes like AM or E.M. Implementing the 'mean mode' postprocessing technique proved highly effective in enhancing the clarity of the system's output and mitigating inaccuracies.

8. CONCLUSION

In this study, we aimed to improve automatic chord recognition using visual data. Our journey from initial ideas to testing the model in practical settings brought out several key insights that contributed to understanding how to build better chord recognition systems. Through well-planned experiments and analyses, we saw the importance of data diversity, tested the use of transfer learning, balanced between image quality and computational demands, and compared the challenges of visual and audio inputs. The findings here show that visual data can be helpful for chord recognition and suggest that combining visual and audio methods could be a good direction for future work. As we discuss the main conclusions and possible next steps in the following sections, our goal remains to work towards a reliable, real-time chord recognition system that performs well in different practical scenarios.

1. Preference for Data Diversity over Volume: Employing a large yet homogeneous dataset does not favorably impact the accuracy of the DCNN model and might pave the way for overfitting. Our trials exhibit that a 25% diminution in the dataset, aided by data augmentation, leads to notable accuracy enhancement.
2. Insights from Transfer Learning: In the fretboard detection workflow, transfer learning emerges as beneficial owing to pre-trained networks with domain relevance. Conversely, transfer learning furnishes limited advantages within the chord classification workflow, as pre-trained networks like ImageNet and COCO lack domain specificity.
3. Trade-offs between Quality and Computation: Increased image quality corresponds positively with model accuracy, yet it levies computational demands, calling for improved hardware resources. Our model attained an optimal accuracy quotient of 83.2% despite hardware constraints.
4. Comparative Difficulties of Visual and Audio Input: The complexities inherent in visual input are similar to those encountered in audio input, particularly when considering the high timbre in audio and the presence of analogous chord patterns in visual data.

9. FUTURE WORKS

While the model displays accuracy, a scope for refinement exists. This investigation affirms that visual data is a hopeful avenue for chord recognition. Future endeavors should focus on:

1. Data Enrichment: A need exists for more diversified data, primarily encompassing different guitar types and lighting scenarios.
2. Multi-Modality: Although visual methods hold promise, amalgamating them with audio methods could potentially elevate the system's efficacy. Following the path of our previous research utilizing multimodal from sound and visual [49]

AUTHOR CONTRIBUTIONS

Conceptualization, Y.K., and L.Z.; methodology, Y.K.; software, M.T.; validation, Y.K., and L.Z.; formal analysis, Y.K, A.J., and M.T.; investigation, Y.K., L.Z., and M.T.; data curation, M.T.; writing—original draft preparation, Y.K., and M.T; writing—review and editing, Y.K., L.Z., and A.J.; visualization, M.T., A.J., and L.Z.; supervision, Y.K. and L.Z.; funding acquisition, A.J. All authors have read and agreed to the published version of the manuscript.

References

[1] J. Pauwels, K. O'Hanlon, E. Gómez, and M. B. Sandler, "20 Years of Automatic Chord Recognition From Audio," *Proceedings of the 20th International Society for Music Information Retrieval Conference, ISMIR 2019*, pp. 54–63, 2019.

[2] T. Fujishima, "Real-time Chord Recognition of Musical Sound: A System Using Common Lisp Music," *ICMC Proceedings*, vol. 9, no. 6. pp. 464–467, 1999.

[3] A. Sheh and D. P. W. Ellis, "Chord segmentation and recognition using EM-trained hidden Markov models," *Proc. ISMIR*, pp. 185–191, 2003.

[4] A. M. Stark and M. D. Plumbley, "Real-time chord recognition for live performance," *Proceedings of the 2009 International Computer Music Conference, ICMC 2009*, no. ICMC, pp. 85–88, 2009.

[5] F. Korzeniowski and G. Widmer, "Feature learning for chord recognition: The deep chroma extractor," *Proceedings of the 17th International Society for Music Information Retrieval Conference, ISMIR 2016*, pp. 37–43, 2016.

[6] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent, "Audio chord recognition with recurrent neural networks," *Proceedings of the 14th International Society for Music Information Retrieval Conference, ISMIR 2013*, pp. 335–340, 2013.

[7] Y. Han, J. Kim, and K. Lee, "Deep Convolutional Neural Networks for Predominant Instrument Recognition in Polyphonic Music," *IEEE/ACM Trans Audio Speech Lang Process*, vol. 25, no. 1, pp. 208–221, 2017.

[8] G. Byambatsogt, L. Choimaa, and G. Koutaki, "Guitar chord sensing and recognition using multi-task learning and physical data augmentation with robotics," *Sensors (Switzerland)*, vol. 20, no. 21, pp. 1–17, 2020.

[9] L. Nanni, A. Rigo, A. Lumini, and S. Brahnam, "Spectrogram classification using dissimilarity space," *Applied Sciences (Switzerland)*, vol. 10, no. 12, pp. 1–17, 2020.

[10] A. Ghriss, "Deep Automatic Chord Recognition," pp. 1–10, 2017.

[11] J. Witte, "AI-technology behind the chords of Chordify – our algorithm explained," *Chordify News*, 2021.

[12] O. Russakovsky *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *Int J Comput Vis*, vol. 115, no. 3, pp. 211–252, 2015.

[13] M. Bajammal, A. Stocco, D. Mazinanian, and A. Mesbah, "A Survey on the Use of Computer Vision to Improve Software Engineering Tasks," *IEEE Transactions on Software Engineering*, no. October 2020.

[14] A.-M. Burns, "Computer Vision Methods for Guitarist Left-Hand Fingering Recognition," 2007.

[15] C. Kerdvibulvech and H. Saito, "Vision-based guitarist fingering tracking using a Bayesian classifier and particle filters," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 4872 LNCS, pp. 625–638, 2007.

[16] C. Kerdvibulvech, "Markerless guitarist fingertip detection using a bayesian classifier and a template matching for supporting guitarists," *Proceedings of the 10th ACM/IEEE Virtual Reality International Conference, VRIC '08*, Laval, France, pp. 2–8, 2008.

[17] J. Scarr and R. Green, "Retrieval of guitarist fingering information using computer vision," *International Conference Image and Vision Computing New Zealand*, pp. 0–6, 2010.

[18] Z. Wang and J. Ohya, "Tracking the guitarist's fingers as well as recognizing pressed chords from a video sequence," *I.S. and T International Symposium on Electronic Imaging Science and Technology*, pp. 1–6, 2016.

[19] M. H. Purnomo, Y. Kristian, E. Setyati, U. Delfana Rosiani, and E. I. Setiawan, "Limitless possibilities of pervasive biomedical engineering: Directing the implementation of affective computing on automatic health monitoring system," in *Proceedings of 2016 8th International Conference on Information Technology and Electrical Engineering: Empowering Technology for Better Future, ICITEE 2016*, 2017.

[20] T. Ooaku, T. D. Linh, M. Arai, T. Maekawa, and K. Mizutani, "Guitar chord recognition based on finger patterns with deep learning," *ACM International Conference Proceeding Series*, pp. 54–57, 2018.

[21] L.-V. Tran, S. Zhang, and E. Zhou, "CNN Transfer Learning for Visual Guitar Chord Classification," 2019. [Online]. Available: `https://api.semanticscholar.org/CorpusID:215776004`

[22] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J Field Robot*, vol. 37, no. 3, pp. 362–386, 2020.

[23] X. Wang, L. M. Meng, B. Zhang, J. Lu, and K. L. Du, "A video-based traffic violation detection system," *Proceedings - 2013 International Conference on Mechatronic Sciences, Electric Engineering and Computer, MEC 2013*, no. December 2013, pp. 1191–1194, 2013.

[24] T. Tirtawan, E. K. Susanto, P. C. S. W. Lukman Zaman and Y. Kristian, "Batik Clothes Auto-Fashion using Conditional Generative Adversarial Network and U-Net," *2021 3rd East Indonesia Conference on Computer and Information Technology (EIConCIT)*, Surabaya, Indonesia, pp. 145-150, 2021.

[25] D. Widjojo, E. Setyati, and Y. Kristian, "Integrated Deep Learning System for Car Damage Detection and Classification Using Deep Transfer Learning," in *2022 IEEE 8th Information Technology International Seminar (ITIS)*, pp. 21–26, 2022.

[26] T. Y. Lin *et al.*, "Microsoft COCO: Common objects in context," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 8693 LNCS, no. PART 5, pp. 740–755, 2014.

[27] A. U. Khan and A. Borji, "Analysis of Hand Segmentation in the Wild," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 4710–4719, 2018.

[28] W. Liu *et al.*, "SSD: Single Shot MultiBox Detector," in *Computer Vision – ECCV 2016*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Cham: Springer International Publishing, 2016, pp. 21–37.

[29] L. Beyer, O. J. Hénaff, A. Kolesnikov, X. Zhai, and A. van den Oord, "Are we done with ImageNet?," 2020.

[30] D. Impiombato *et al.*, "SSD: Single Shot MultiBox Detector Wei," *Nucl Instrum Methods Phys Res A*, vol. 794, pp. 185–192, 2015.

[31] A. G. Howard *et al.*, "MobileNets: Efficient Con-

volutional Neural Networks for Mobile Vision Applications," 2017.

[32] A. G. Howard *et al.*, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *ArXiv*, vol. abs/1704.04861, 2017, [Online]. Available: `https://api.semanticscholar.org/CorpusID:12670695`

[33] F. Chollet, "Xception: Deep learning with depthwise separable convolutions," *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition*, CVPR 2017, vol. 2017-Janua, pp. 1800–1807, 2017.

[34] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, pp. 2818–2826, 2016.

[35] C. Szegedy *et al.*, "Going deeper with convolutions," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 07-12-June, pp. 1–9, 2015.

[36] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *32nd International Conference on Machine Learning, ICML 2015*, vol. 1, pp. 448–456, 2015.

[37] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi, "Inception-v4, inception-ResNet and the impact of residual connections on learning," *31st AAAI Conference on Artificial Intelligence, AAAI 2017*, pp. 4278–4284, 2017.

[38] Y. Guo, Y. Liu, A. Oerlemans, S. Lao, S. Wu, and M. S. Lew, "Deep learning for visual understanding: A review," *Neurocomputing*, vol. 187, pp. 27–48, 2016.

[39] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," *Commun. ACM*, vol. 60, no. 6, pp. 84–90, 2017.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 37, no. 9, pp. 1904–1916, Sep. 2015.

[41] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[42] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[43] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," *arXiv:1412.6980*, 2017.

[44] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[45] Y. You *et al.*, "Large Batch Optimization for Deep Learning: Training BERT in 76 minutes," *arXiv:1904.00962*, 2020.

[46] C. Shorten and T. M. Khoshgoftaar, "A survey on Image Data Augmentation for Deep Learning," *J Big Data*, vol. 6, no. 1, p. 60, 2019.

[47] A. Mikolajczyk and M. Grochowski, "Data augmentation for improving deep learning in image classification problem," in *2018 International Interdisciplinary PhD Workshop (IIPhDW)*, pp. 117–122, 2018.

[48] L. Perez and J. Wang, "The Effectiveness of Data Augmentation in Image Classification using Deep Learning," *arXiv:1712.04621*, 2017.

[49] Y. Kristian, N. Simogiarto, M. T. A. Sampurna, E. Hanindito, and V. Visuddho, "Ensemble of multimodal deep learning autoencoder for infant cry and pain detection," *F1000Res*, vol. 11, p. 359, 2023.

Yosi Kristian was born in Tuban, East Java, Indonesia, in 1981. He obtained his bachelor's degree in Computer Science and master's degree in Information Technology from Sekolah Tinggi Teknik Surabaya (now known as the Institut Sains dan Teknologi Terpadu Surabaya or Institut STTS) in 2004 and 2008, respectively. In 2018, he earned his Ph.D. from the Institut Teknologi Sepuluh Nopember (ITS) in Surabaya, Indonesia. In 2015, he was a research student at Osaka City University. Since 2004, Yosi has been a faculty member at Institut STTS, currently serving as an Associate Professor and Head of the Department of Informatics. His research interests include Machine Learning, Deep Learning, Artificial Intelligence, and Computer Vision.

Lukman Zaman earned his undergraduate degree in computer science from Sekolah Tinggi Teknik Surabaya (now known as the Institut Sains dan Teknologi Terpadu Surabaya or Institut STTS), Surabaya, Indonesia, in 1998. He then obtained his master's degree in Informatics from Institut Teknologi Sepuluh Nopember (ITS), Surabaya, Indonesia, in 2003. He has recently completed his doctoral degree at the same institution. Since 1999, he has been a faculty member at Institut STTS. His research interests are interactive media, generative artificial intelligence, and computer graphics.

Michael Tenoyo was born in 1999 in Surabaya, Indonesia. He graduated with a bachelor's degree in Informatics from ISTTS in 2021. Michael is passionate about developing and researching artificial intelligence implementations for daily use and problem-solving. He has worked as a full-stack programmer and software engineer in Tokyo, Japan, since early 2024.

Andreas Jodhinata (born 01-12-1968) completed his undergraduate and master's degrees at Sekolah Tinggi Teknik Surabaya (now known as the Institut Sains dan Teknologi Terpadu Surabaya or Institut STTS) and his Ph.D. at Institut Teknologi Sepuluh Nopember (ITS). In 2016, he had the opportunity to go to Tokyo University of Technology to conduct research related to machinima. He was Head of the Information Systems Study Program at Universitas Pelita Harapan Surabaya for 10 years. Currently, he serves as Head of the Online Informatics Program at Ciputra University Surabaya. He is interested in Machine Learning, Artificial Intelligence, and Computer Vision.