Pose Estimation for Musical Exercise

Group number:
313

Group members details:

David Shavin
208925065
david.shavin@mail.huji.ac.il

David Roskin
317491868
David.Roskin@mail.huji.ac.il

Advisor details:
Gal Katzhendler
gal.katzhendler@mail.huji.ac.il
Hebrew University Jerusalem Israel

Mentor name:
Guy Hacohen

Abstract

Pose estimation is a computer vision task in which information is retrieved from images about a pose of an entity. Its implementations are seen in gaming, camera security surveillance, etc. In recent years there has been tremendous progress in the field thanks to the advancement in deep learning. Our project aims to use computer vision and audio processing to help beginner guitarists to learn to play the guitar. To solve this problem, we trained a deep learning model, that uses visual and audio data to classify the chord and give feedback to the user. The main challenge of our project was to deal with the various hand forms in which different people are playing the same chords and the similarity between hand forms of different chords. Our final product is an app in which users will record themselves playing their favorite songs and receive live feedback according to their playing.

1 Introduction

Since the advent of the internet, more people are referring to online lessons to learn new skills on platforms like Youtube, Coursera, and more. It is common for beginner musicians to use these platforms to learn a new instrument by themselves, however, many struggle to figure out if their playing is correct, and therefore, they require a human teacher to assess them.
In this paper we intend to solve this problem for beginner guitar players by offering a computer program that can record the user play and give live feedback on his playing. When using our program the user will be able to see if his playing is correct and where he needs to improve.
The task of classifying the chords is challenging since there are many different chords, some of which are very similar both visually (the hand's pose on the guitar-bar) and in sound. Guitar chords are usually produced when strumming on 4-6 strings simultaneously. The different strings have different levels of amplitude and different harmonics. For that reason the naive approach of, transforming the sound to its Fourier domain, extracting the different notes by taking the dominant frequencies and restoring from it the played chord, fails. This task is even harder in noisy surroundings since most models nowadays rely only on audio to classify the chords, e.g. Chord ai-Real-time chord recognition, SimplyGuitar, etc.
In our solution, we propose to better use the data available to us for the guitar's chord's classifier by taking in as an input to the model both an image of the hand's pose playing the guitar and the sound from the guitar. We'll use a neural network to classify the chords.
Our project is built for beginner guitar players who can now start learning to play the guitar for free and in a fun way.

2 Related Work

In this section, we describe previous work in the domain of images and sound classification and in particular on guitars. We'll discuss work made only on images using computer-vision techniques, work made only on audio signals using signal-processing techniques, and work made using images and audio data in the domain of speech recognition

2.1 Visual Techniques

Before the "deep Learning revolution" in 2012, where AlexNet[1] won the ImageNet competition, most methods for guitar chords classifications were divided into recognition of the guitar bar, strings, and frets and finger recognition. Once done, the classification of the chords was a task of finding which fingers is pressing on which note. Motokawa and Saito, 2006 [2] demonstrated a system of locating the guitar bar using an AR marker however using an AR marker isn't suitable for the wide public. Scarr and Green, 2010 [3] proposed a method of using Canny edge detection and probabilistic Hough transform to locate the strings and frets, however, they require the user to take an image of the scene with the guitar and another one without the guitar, using the subtraction between the two images to create a mask around the guitar, which is inconvenient for the user. In addition, they detected the fingers by detecting the largest contours on the guitar bar[1] after applying Canny edge detection. Zhang et al, 2020 [4] use a ANN for real-time hand landmarks detection, achieving a 95.7% average precision. The weakness of these methods is they assume that if a finger is between 2 frets and over a certain string then the finger is pressing on the string. This type of information isn't always possible to detect, however, observing the hand's pose as a whole can sometimes hint at if the fingers are pressing the string or not.

2.2 Audio Techniques

A significant amount of research has been in audio classification via signal processing. Gunhild Elisabeth Berget, 2017 [5] tackled the problem of generating a chord progression from melody[2]. She proposed to learn the underlying Markov chain chord progression from training data and then adjust the chord progression using melody input. Jesus Guerrero-Turrubiates et al, 2015 [6] proposed a chord classification method for guitar based on feeding the Fourier signal of a guitar's strum to an ANN, achieving an accuracy of 95.3% on a total of 48 different chords. While these methods work well, audio-based models remain sensitive to background noises and don't capture the visual information that is available to most people today via their smartphones.

2.3 Audio & Visual Techniques

When visual and audio events occur together, it suggests that there might be a common, underlying event that produces both signals. Andrew Owens et al, 2018 [7] showed ANNs-based methods of processing audio and visual together for multiple applications, such as sound source localization, audio-visual action recognition, and on/off-screen audio source separation, e.g. removing the off-screen translator's voice from a foreign official's speech. Our system utilizes these concepts to achieve a classifier for guitar chords that is accurate and isn't sensitive to noise.

---

[1] They added a few other morphological criteria to determine if the contour is a finger.
[2] Mostly relevant when playing fingerstyle guitar.

3 Methods & Materials

In our final product, the user inserts a name of a song he wants to play. In the first step, a crawler will extract the lyrics and the chords from Ultimate Guitar[3], displaying it to the user. The program will then record the user, repeatedly processing the last second of the recording, storing the audio signal and the middle frame (the frame at time 0.5 of that second). The program will check if the user is currently playing by measuring the volume it recorded. If the recorded volume is over a certain threshold then it will consider that second as a second in which the player is playing the guitar. The threshold was picked based on empirical results. If the user is currently playing then the program will preprocess the data (see section 3.2) and insert it into the classifier. It then compares the prediction with the chord that is currently to be played and outputs feedback accordingly. If the feedback is positive, the program moves to the next chord in the song, else, it will request the user to play the chord again.

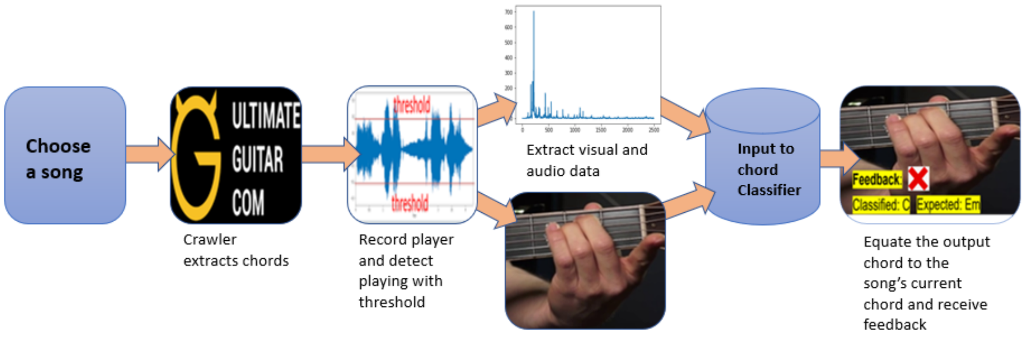

Choose a song
ULTIMATE GUITAR COM
Crawler extracts chords
threshold
threshold
Record player and detect playing with threshold
Extract visual and audio data
Input to chord Classifier
Feedback:
Classified: C
Expected: Em
Equate the output chord to the song's current chord and receive feedback

Figure 1: project pipeline

3.1 Dataset

To have the ability to classify the chords, we needed to create a dataset. Therefore, we recorded ourselves playing different chords and downloaded videos of people playing chords from Youtube. We've added those videos to create a more diverse dataset with different guitar angles, hand poses, lighting, etc. Eventually, we had 1497 videos. These videos contain eleven of the most popular chords, A, Am, B, Bm, C, D, Dm, E, Em, F, and G, which will be used as labels for our classifier. Fortunately, most songs consist of a subset of these chords and often enough, if a chord in the song isn't part of our labels, it might be a variation of a chord that is in our labels, e.g. F7 is just a variation of the chord F. Therefore, replacing a chord that isn't in our labels with a variation of the chord that is in our labels, drastically enlarges the number of songs we're able to fully present for the user. We've limited ourselves to popular chords because there are hundreds of different chords that for the most part don't appear in songs. This makes it difficult to gather data on these chords and makes this task pointless since these chords are rarely used. For each video, we took a second-long clip of a chord being played while the hand pose didn't change. We then extracted a WAV file of the clip's audio and a crop of the hand from the frame at time 0.5. We repeated this procedure, allowing a maximum of ten 1-second clips of each person playing the guitar and thus, prevented the data from being uniform.

3.2 Guitar Chord Classifier

Before inserting data into the model we built, we first had to preprocess the data to our desired format. The preprocessing of the data is divided into two parts: Preprocessing while training the model and preprocessing during real-time playing. The only difference between these processes is that during training, we had to load the data from the dataset.
For processing the audio, we'll first transform the signal to its frequency domain, filtering the frequencies with low amplitude by removing 90% of the softest frequencies[4]. This method is used for filtering noise and is especially useful for our task since we already know guitar chords are built from a relatively small amount of constant frequencies, and thus most of the low amplitude frequencies in the audio signal are background noises. Finally, we normalized the signal. For preprocessing the image, we'll use the hand detector we've implemented using media pipe [4] to detect and crop the hand from the frame we've stored. we'll then resize and normalize the image's intensity as a final stage.
The model we've built for classifying the chord consists of two stages: feature extraction, and classification. In the first stage, we passed the hand's image through a convolutional neural network (CNN) and the processed signal through a multi-layer perceptron (MLP) to extract image features and audio features accordingly. We've then flattened the features and concatenated them, creating a vector of features, which we've then passed through an MLP classifier. The final stage of the classifier consisted of a Softmax, resulting in a vector of probabilities for each class. The classifier's prediction is then the class with the maximal probability. The optimizer we've used is Adam [8], updating the gradients with a learning rate of 3e-4. The different learning rates we've used didn't change much, however, 3e-4 usually gave the best results. For the loss function, we've used a Cross Entropy Loss. The architecture of the model can be seen in figure 2.

[3] Ultimate Guitar is a popular website, containing a large database of songs labeled with their guitar chords.
[4] Empirically, removing 90% of the signal gives good results.

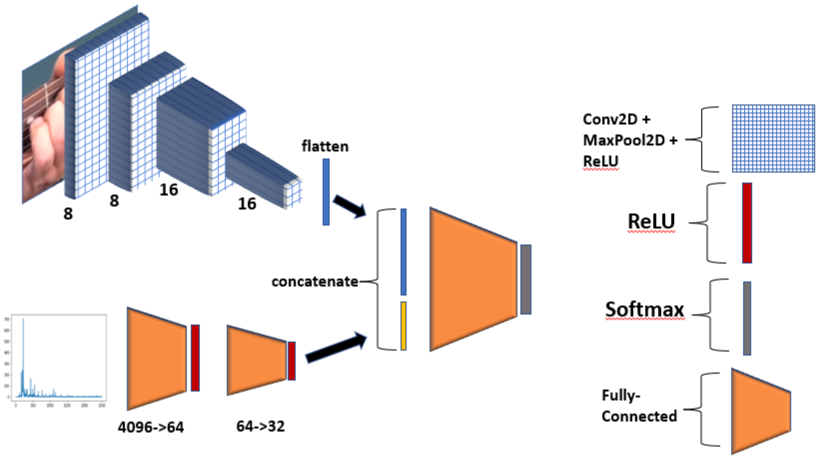


Figure 2: Model Pipeline, kernel size=3, pooling=2x2

4 Evaluation & Verification

Evaluating our project properly can only be done by evaluating each part of the pipeline separately. We therefore validated each part of the project in a few different ways, ranging from accuracy to runtime performance. For Evaluating the chord classifier model we've used a validation set of 30 samples per class resulting in 11x30=330 samples.

4.1 Visual Model, Accuracy vs. Runtime

In our work, we presented the use of a CNN for visual feature extraction. For our program to work on a standard CPU we had to choose an architecture that can successfully classify chords and even more importantly has a low running time. To measure the running time of a model, we measured the time it took from the moment we inserted an image into the model till the moment the model outputted the result. In figure 3 we can see a scatter plot displaying the accuracy vs. running time for different CNN models[5]. The models were validated on our validation set, and the time was measured as the mean time of 1000 classification, running on an Intel(R) Core(TM i5-7200U CPU @ 2.50GHz, 2712 Mhz 2 Core(s). 4 Logical Processor(s).

The plot shows that while the different models have a relatively small impact on the accuracy, the running time is clearly in favor of our custom CNN model. This is because our model is much smaller, consisting of only a few layers and overall of 316931 parameters which are far less than the other models that consist of millions of parameters.

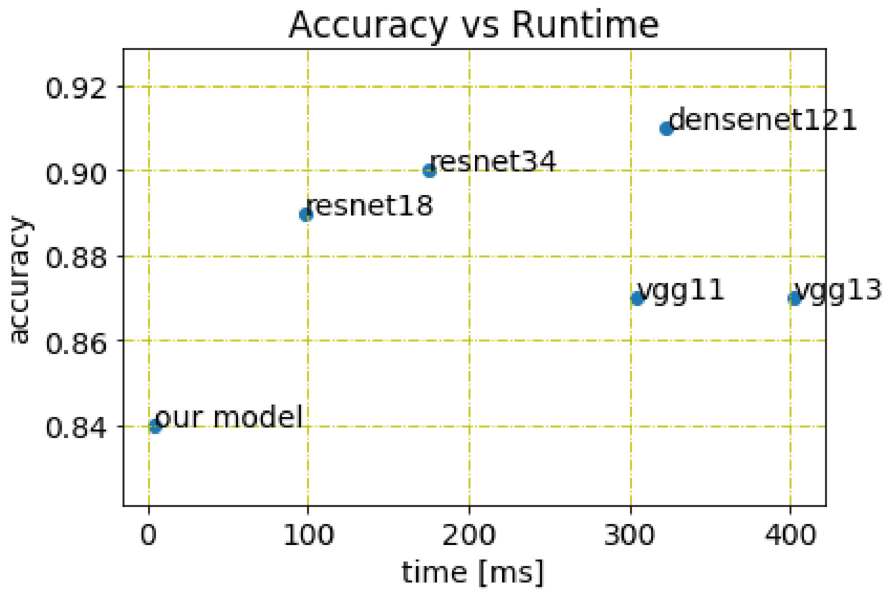


Figure 3: Accuracy Vs. Runtime

4.2 Visual vs. Audio vs. Audio & Visual

We've built and evaluated three different deep-learning models: Relying solely on audio, solely on visual, and on both.

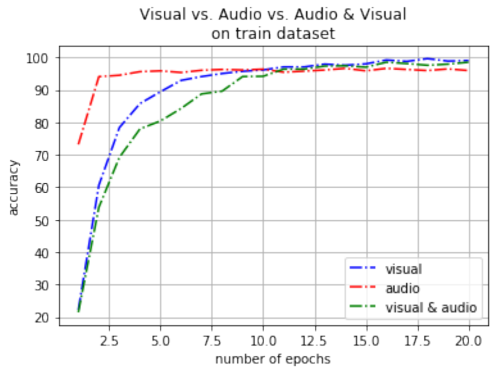


Figure 4: Model comparison, train dataset

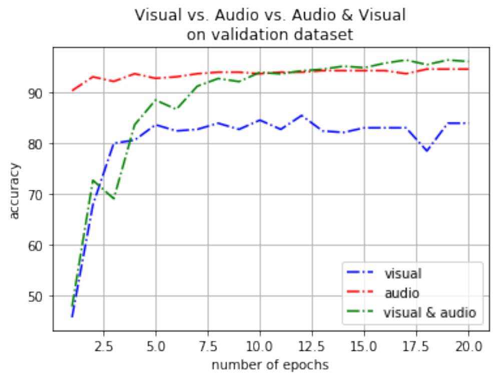


Figure 5: Model comparison, validation dataset

Taking a look at the results we can see that both the audio model and the visual-audio model enjoyed greater success than the visual model. The reason for that difference could be explained by the nature of chords. Chords consist of a combination of frequencies played at the same time. Therefore, given the frequencies of an audio signal, a learning model can learn roughly which frequencies construct each chord and classify the chords according to the frequencies it consists of. This task is relatively simple, and so it's no surprise that the audio model was able to achieve good results relatively fast. Unlike the task of chord classification via audio, the task of chord classification via images is less straightforward. The variance of different hand poses each chord has is quite big, and many different chords share similar hand poses. It's interesting to note that the model that used both audio signals and images as input has managed to outperform the audio model. This shows that the model has managed to learn connections between hand poses and the sound of their corresponding chords. This is not a trivial task considering there's no direct connection between the images domain and the sound domain.

4.3 Chord Classification ROC Curves

In our project, it was important for us to get a good AUC in our ROC curves. Since our project deals with multi-class classification, we've adapted the one-vs-rest[6] (ovr) approach when building our ROC curves. We've built a ROC curve for each class and added the confusion matrix in order to show the full picture of our model's classifications on our validation set.

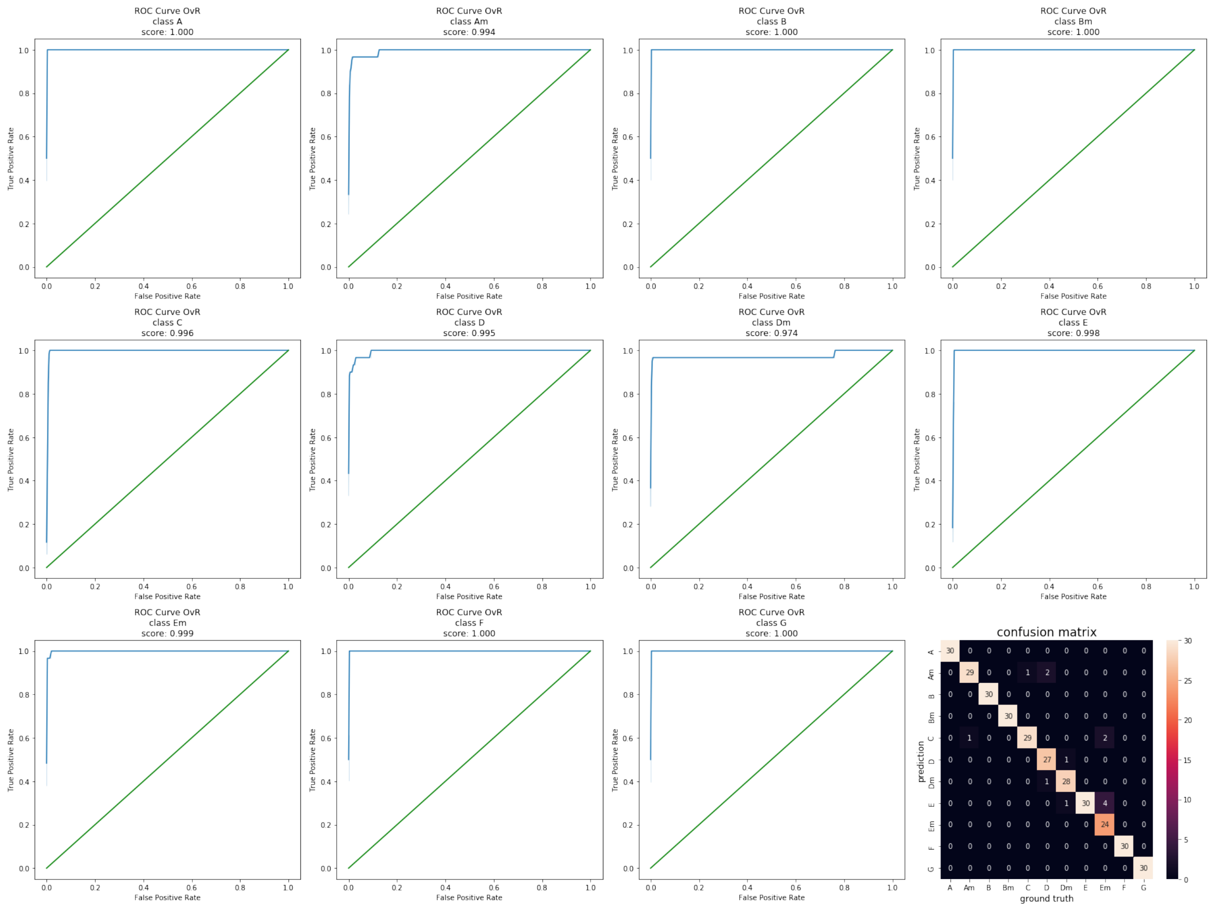


Figure 6: ROC graphs OVR & confusion matrix

4.4 Hand Detection Validation

To detect the hand's pose, we've used the Media-Pipe[4] python library. Their model is based on two CNN architectures: A palm detector model and a hand landmark model, achieving 95.7% average precision.

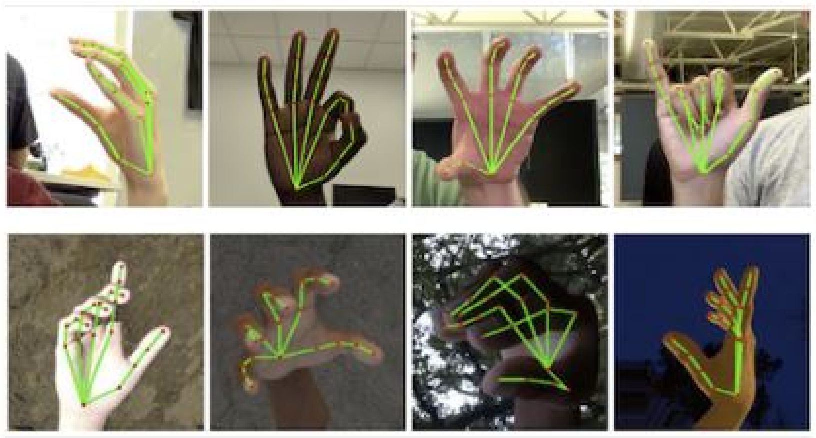


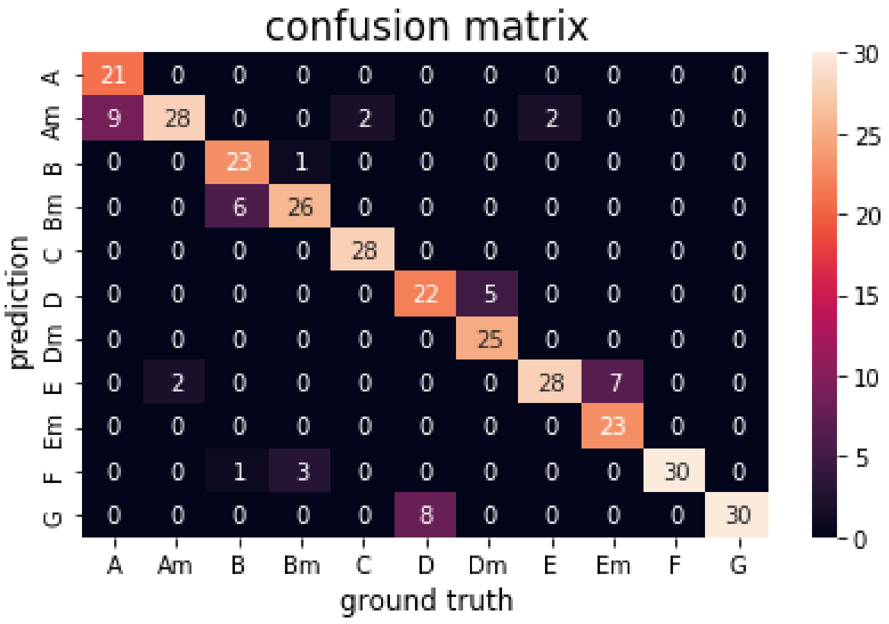
Figure 7: Examples of hand landmarks. (Top): Annotated real-world images. (Bottom): Rendered synthetic hand images with ground truth annotation

4.5 Crawler Validation

The crawler we've built to extract the lyrics and chords from the web successfully extracted the correct songs 96.6% of the time. The crawler failed only in cases of ambiguity regarding the name of the song. For example, if there are other songs with the same name. The crawler's performance was evaluated on 30 songs, all in English.

4.6 Project Verification

To verify our project, we tested our program in three different environments on three different people. Each person played each chord ten times. We present the results in figure 8.



prediction \ ground truth	A	Am	B	Bm	C	D	Dm	E	Em	F	G
A	21	0	0	0	0	0	0	0	0	0	0
Am	9	28	0	0	2	0	0	2	0	0	0
B	0	0	23	1	0	0	0	0	0	0	0
Bm	0	0	6	26	0	0	0	0	0	0	0
C	0	0	0	0	28	0	0	0	0	0	0
D	0	0	0	0	0	22	5	0	0	0	0
Dm	0	0	0	0	0	0	25	0	0	0	0
E	0	2	0	0	0	0	0	28	7	0	0
Em	0	0	0	0	0	0	0	0	23	0	0
F	0	0	1	3	0	0	0	0	0	30	0
G	0	0	0	0	0	8	0	0	0	0	30

Figure 8: Confusion-matrix verification

While the results we obtained on the validation dataset were promising, in reality our model didn't perform as well. A big reason for this is the quality of the microphone we used and the camera position. When verifying our project, we tested the program in real live situations with someone playing guitar. This is in contrast to the dataset that we used to train and validate our model, which was built from tutorial videos uploaded to the web. These videos were captured in a studio with a fixed high-quality camera and microphone. It is therefore not surprising that our verification results weren't as good.

[5] The fully connected layer at the end of these models had to be changed to fit the number of classes in our task.
[6] See https://towardsdatascience.com/multiclass-classification-evaluation-with-roc-curves-and-roc-auc-294fd4617e3a for more information

5 Conclusions and Future Work

In our project we managed to get the results we initially hoped for. Unfortunately, our model is limited to classification based on a 1-second recording. This requires the user to play relatively slow and is therefore not usable by professional guitar players.
Due to limitations in the dataset, we weren't able to fully measure the impact of building a model that uses both audio and visuals to predict guitar chords vs. a model that uses only audio. This is because the accuracy of our audio model and our audio-visual model were too close to justify the need for visual data (images). To better test the necessity of visual data for chord classification, a richer dataset is needed.
A source of inspiration for our audio-visual model was the cocktail party problem[7]. It is well known that there is no easy way to associate each separated audio source with its corresponding speaker in a video, Ephrat et al, 2018 [9] have shown that using the available visual data of people speaking as well as the audio data in an ANN, can drastically improve the results of noise separation. In our project, we've implemented that approach with guitar, and in future work we believe that similar concepts can be used to separate the guitar from background noises, and even separate the guitar from other instruments.

[7]The "cocktail party problem" is encountered when sounds from different sources in the world mix in the air before arriving at the ear, requiring the brain to estimate individual sources from the received mixture.

6 Acknowledgements

We would like to thank the Hebrew University engineering staff for guidance and resources, and in particular we would like to thank Mr. Guy Hacohen for mentoring us through this project.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in neural information processing systems*, vol. 25, 2012.

[2] Y. Motokawa and H. Saito, "Support system for guitar playing using augmented reality display," in *2006 IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE, 2006, pp. 243–244.

[3] J. Scarr and R. Green, "Retrieval of guitarist fingering information using computer vision," in *2010 25th International Conference of Image and Vision Computing New Zealand*. IEEE, 2010, pp. 1–7.

[4] F. Zhang, V. Bazarevsky, A. Vakunov, A. Tkachenka, G. Sung, C.-L. Chang, and M. Grundmann, "Mediapipe hands: On-device real-time hand tracking," *arXiv preprint arXiv:2006.10214*, 2020.

[5] G. E. Berget, "Using hidden markov models for musical chord prediction," Master's thesis, NTNU, 2017.

[6] J. Guerrero-Turrubiates, S. Ledesma, S. Gonzalez-Reyna, and G. Avina-Cervantes, "Guitar chords classification using uncertainty measurements of frequency bins," *Mathematical Problems in Engineering*, vol. 2015, 2015.

[7] A. Owens and A. A. Efros, "Audio-visual scene analysis with self-supervised multisensory features," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[8] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[9] A. Ephrat, I. Mosseri, O. Lang, T. Dekel, K. Wilson, A. Hassidim, W. T. Freeman, and M. Rubinstein, "Looking to listen at the cocktail party: A speaker-independent audio-visual model for speech separation," *arXiv preprint arXiv:1804.03619*, 2018.