Live Hand Pose Detection and Music Generation via Video Feed

Team Name : Pipeline Predators
CSE518: Artificial Intelligence — Mid-semester Report

Kirtan Kalaria
AUL202005
Ahmedabad University
Ahmedabad, India
kirtan.k@ahduni.edu.in

Sanya Zaveri
AU1920064
Ahmedabad University
Ahmedabad, India
sanya.z@ahduni.edu.in

Riddhi Parekh
AU1920100
Ahmedabad University
Ahmedabad, India
riddhi.p3@ahduni.edu.in

Foram Shah
AU1920185
Ahmedabad University
Ahmedabad, India
foram.s1@ahduni.edu.in

CONTENTS

***Abstract*—In this project, our objective is to extract hand pose by the way of 2D coordinated of keypoints from a live video/webcam feed. Our approach is CNN based, and we are training it to estimate 2D coordinates from 2D RGB images.**

***Index Terms*—Hand pose, computer vision, CNN, regression, keypoints, skeletal annotation.**

I. INTRODUCTION

Contact-less control has become a fundamental feature of recent and emerging human-interaction-based technology. Especially in AR and VR, where true physical interaction in the virtual or projected environment is impossible, various modalities like hand gestures and speech are utilized to bridge the gap for natural intuitive interaction. In this project, our objective is to design a system that detects the hand pose from a live camera feed, and performs the application of generating music dynamically based on the detected hand pose, much like the musical instrument theremin.

II. LITERATURE SURVEY

In recent years, there have been multiple studies on hand pose detection models. This detection has served many purposes including hand tracking, hand parsing, fingertip detection, joints estimation, hand contour estimation, gesture recognition, etc. The main factors affecting these detections are low resolution, self similarity, incomplete data, annotation difficulties, real-time performance, etc. Generally, hand pose detection methods are model-based method, appearance-based method, and hybrid method; some are reviewed in [1] and [2]. Currently, CNN based approaches are one of the most accurate and successful ones in the detection and recognition of the hand pose.

Insight of Model based approach is first undertaken using a hand model which provides a robust, coarse estimate which is followed by a fine-tuning approach used to improve estimation accuracy which is reviewed in [3]. A 3D CNN-based hand pose estimation approach that can capture the 3D spatial structure of the input and accurately regress full 3D hand pose in a single pass is given in [4]. Paper [4] focuses upon the 3D shallow plain network and deep dense network and how it achieves superior accuracy and avoids the time consuming refinement process. Another good survey of hand pose estimation is shown in [5] where the author discusses 3D hand pose estimation by CNNs to estimate 3D hand pose from some types of data, including the depth image, the RGB image , the RGB-D image, or other camera data and discusses some results of 3D hand pose estimation by the CNNs. We restrict our work to 2D hand pose.

III. IMPLEMENTATION

A. Dataset

After considering several datasets, we decided to use the Large-scale Multiview 3D Hand Pose Dataset, because it satisfied a few requirements we had.

- It contains RGB images.
- It is publicly available and free to use.
- It is a very large dataset.
- It has 22 keypoints including the palm normal.
- It has both 2D and 3D coordinates for the keypoints.
- It has the views from 4 different angles for each pose, hence there is good variation in the data.
- The authors of the dataset have included physical variation such as the presence and absence of rings and gloves, different peoples' hands (skin color, shape, etc.), and left and right sides.
- Moreover, they have artificially augmented data as well, with many virtual backgrounds, so the CNN can be trained for a variety of user backgrounds.

B. Pre-processing

From the 3D coordinates provided for the dataset, such that for every set of 4 images describing the same pose from different cameras at different angles, the coordinates of the keypoints remain the same, and in the frame of the Leap Motion controller. So, the first thing we needed to do is perform projection transitions to get the 2D coordinates for each image in their own frame of reference.

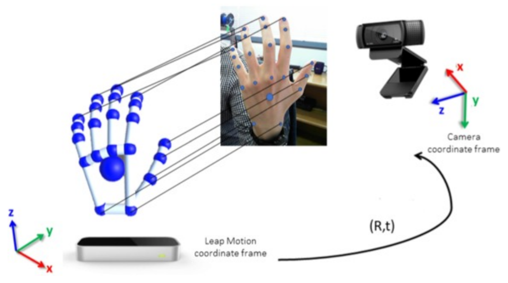

Fig. 1. Projecting coordinates to image frames

Then we had to generate a Pandas DataFrame consisting of image paths and corresponding newly generated x and y coordinates for each keypoint. We then take a random sample of it to use, to make training computationally feasible. Finally, we split it into train-validation data and make respective Keras ImageDataGenerators to pass to the model for training. ImageDataGenerators were indispensable because otherwise the input array consisting of the image arrays was extremely large and was computationally impossible to make with the available resources. We did not use the dataset to do the testing. Rather, we tested on the live webcam feed.

C. CNN Architecture

After hyperparameter optimization, we fixed on a suitable model architecture that balanced the bias against the variance. We chose to use our custom model instead of standard models. However, VGG-19 was the basis of our inspiration.

Layers:

1) Input (RGB Images, 3 channels, pixels rescaled to [0,1])
2) 2 × Convolution (64 filters, 3×3 kernel, ReLU)
3) MaxPooling (/2)
4) 3 × Convolution (128 filters, 3×3 kernel, ReLU)
5) MaxPooling (/2)
6) 5 × Convolution (256 filters, 3×3 kernel, ReLU)
7) MaxPooling (/2)
8) 5 × Convolution (512 filters, 3×3 kernel, ReLU)
9) MaxPooling (/2)
10) 5 × Convolution (512 filters, 3×3 kernel, ReLU)
11) MaxPooling (/2)
12) Flatten [32768 units]
13) Dense (4000, ReLU)
14) Droupout(0.2)
15) Dense (1000, ReLU)
16) Droupout(0.2)
17) Dense (500, ReLU)
18) Dense (100, ReLU)
19) Dense (42, Relu, Output)

D. Training

We train the CNN to estimate keypoint locations from hand images.

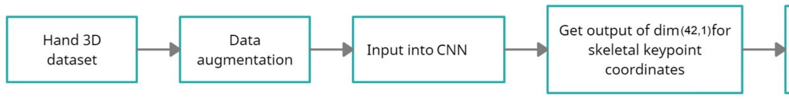

Fig. 2. Training Process

E. Testing

We test the model on live video feed by estimating keypoint locations on individual frames.

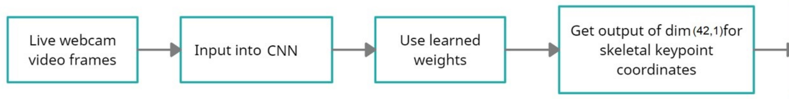

Fig. 3. Training Process

F. Music Generation

We used a simple mathematical function to map the coordinates to a note.

- Root mean square of all 21 x coordinates was calculated.
- It was squared.
- Then it was multiplied once by 2000 and once by 4000 to yield two different frequencies (say, frequency 1 and frequency 2).
- Frequency 1 was used to create a sine wave note and frequency 2 was used to create a cosine wave note.

Similar exercise was done with the y coordinates. All in all, we had 4 notes (waves). We added all of them to get the final wave note to play.

If R_X is the root mean square of all x coordinates and R_Y is the root mean square of all y coordinates, the equation of the wave describing the note played is:

$$sin(2000R_X^2t)+cos(4000R_X^2t)+sin(2000R_Y^2t)+cos(4000R_Y^2t) \tag{1}$$

.

Our objective was to create a relatively complex and pleasant wave (for quality) rather than a simple sine wave that produced an unpleasant beep, while maintaining simplicity for the user to exercise control over their music generation. Apart from squaring the root mean square, the complexity of the function also helps increase the range of producible music.

G. Issues

The primary issues we faced are related to computing power limitations. The RAM on Google Colab and Kaggle Notebooks runs out. Using alternative approaches, the speed of execution becomes painfully slow. The GPU time allowed is also insufficient. However, sampling of the datast and using ImageDataGenerator from Keras, alleviated most of the computational restrictions.

IV. Results

A. Hand Pose Results

Here are a few results of testing with the webcam. The hand pose skeleton was plotted over the frame images using the coordinates of keypoints.

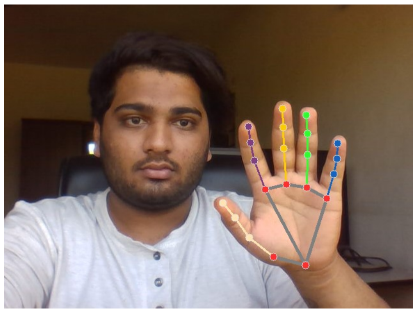

Fig. 4. Inner side of hand

As we can see, the results are mostly satisfactory. From Fig. 6, Fig. 7, and Fig. 8, we can see that despite some slightly off-target predictions, the model predicts keypoints reasonably well even in difficult cases. However, one of our limitations is that currently, we have designed the pipeline to work only on a single hand. With some work, it can be extended to work on multiple hands. Moreover, sometimes there is are time lags but usually the performance is satisfactory.

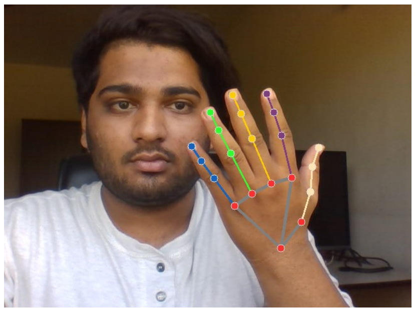

Fig. 5. Outer side of hand

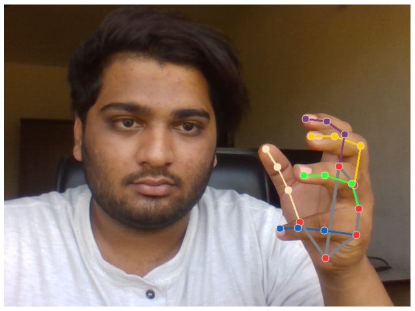

Fig. 6. Curled hand and twisted skeleton

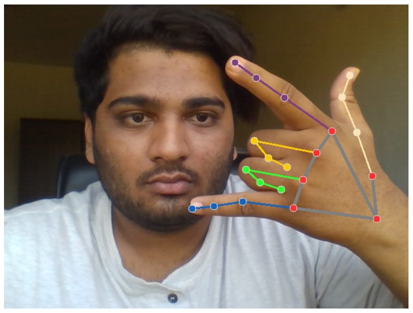

Fig. 7. Curled fingers and prediction on hidden part of fingers

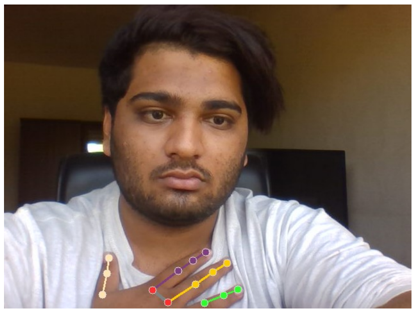

Fig. 8. Prediction on partial hand

B. Music Generation

The music generation part works well and the note frequencies are responsive to changes in the hand pose or location within the frame. The results can not be displayed here on paper.

From the point of view of the program as an instrument, a very slight change in the hand pose or location of hand changed the frequency slightly but noticeably, but did not changed the waveform much. However, a larger change in the location changed both the waveform and the frequency. The same happens with the hand pose but it depends on how the hand pose changes the root mean square changes.

Overall, the music generation depends explicitly on the mathematical function used to produce it from the hand pose and thus is an independent application based on but not dependant on the hand pose detection system.

V. Explainable AI

Despite their success in various AI tasks and the resulting popularity, Convolutional Neural Networks are notorious for being black boxes. How CNNs learn is easy to understand and control. However, understanding what they are learning is a puzzling task even for the experienced. Here is where visualizing feature maps come in. To get a closer look on what type of features our network learns, we visualize the activated output (feature maps) of every convolutional layer in our network. Some of them are given in Fig. 9 to 11.

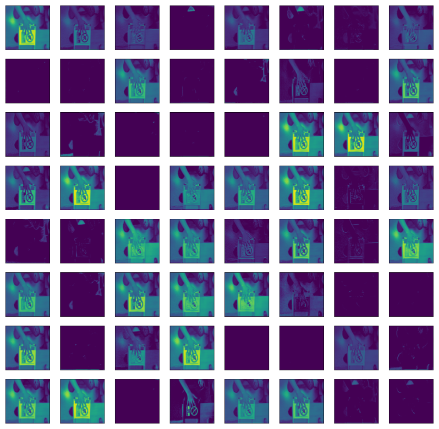
Fig. 9. Feature maps from output of 1^{st} convolutional layer

We min-max normalized the feature maps pixels and them multiplied them by 255, essentially normalizing them back to [0,255].

We can directly observe that the initial layers of the network detect low-level features (colours, edges, etc.) while the later layers of the network detect high-level features (shapes and objects). For any input, we can observe these features and relate it with the performance of the model on that input. For poor prediction, we can observe what features are learned incorrectly. Usually, it is the high-level features that seem to be the issue, as the low-level features are correct as long as the weights were randomized before training. An inability

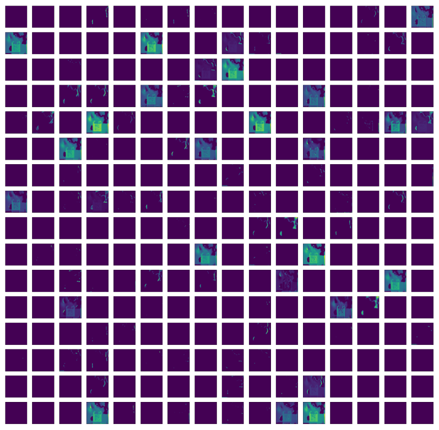
Fig. 10. Feature maps from output of 10^{th} convolutional layer

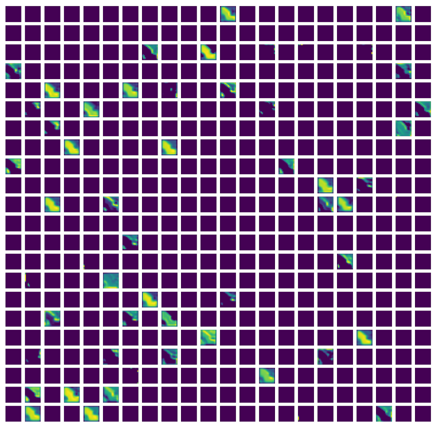
Fig. 11. Feature maps from output of 20^{th}(last) convolutional layer

to learn the higher level features is an indication that the model complexity is insufficient or the type of architecture is improper. More specifically, higher level feature maps display the capacity of models to focus on big and small regions of interest in the input.

This, coupled with training and validation metrics and loss data, helped us immensely optimize our hyperparameters and model architecture.

VI. Conclusion

We have been able to perform the hand pose estimation task on live video input, despite facing some computational challenges. Moreover, we have successfully implemented live music generation based on the live hand pose. Displaying the hand pose skeleton and playing the music takes place simultaneously. Despite satisfactory results, there is room for improvement.

References

[1] Ali Erol, George Bebis, Mircea Nicolescu, Richard D. Boyle, Xander Twombly, "Vision-based hand pose estimation: A review", 2007. [Online]. Available: https://doi.org/10.1016/j.cviu.2006.10.012. [Accessed: 19-Oct-2021].

[2] E. Barsoum, “Articulated Hand Pose Estimation Review,” 2016. [Online]. Available: https://arxiv.org/pdf/1604.06195. [Accessed: 19-Oct-2021].

[3] Lu Ding, Yong Wang, Robert Laganiere, Dan Huang, Shan Fud, ”A CNN model for real time hand pose estimation”, August 2021. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S1047320321001279. [Accessed: 19-Oct-2021].

[4] Liuhao Ge, Hui Liang, Junsong Yuan, Daniel Thalmann, ”Real-Time 3D Hand Pose Estimation with 3D Convolutional Neural Networks”, April 2016. [Online]. Available: https://pubmed.ncbi.nlm.nih.gov/29993927/. [Accessed: 19-Oct-2021].

[5] Van-Hung Le, Hung-Cuong Nguyen, ”A Survey on 3D Hand Skeleton and Pose Estimation by Convolutional Neural Network”, July 2020. [Online]. Available: https://astesj.com/v05/i04/p18/. [Accessed: 19-Oct-2021].