HAND GESTURES RECOGNITION

SUBMITTED TO

ANKITA WADHWANI MAM

BY:

Arpit Banerjie 11806194

Shaik Riyaz 11805957

Rishab Agarwal 11806342

INTRODUCTION

This project was started with a vision of providing a modular approach to build a hand gesture recognition program. Over the years computers have often taken inputs from keyboard and mouse in order to perform basic functions or to let the users interact with their computers and even though computers do take video as input through the webcam it has never been a meaningful way of performing any task on the pc itself. We however try to make it useful as this project allows us to detect various hand gestures in order to perform some tasks by using those gestures as commands.

LITERATURE REVIEW

S. No.	Research Paper	Year
1	Hand gesture recognition using multilayer perceptron network	2015
2	Static hand gesture recognition based on convolutional neural networks	2019
3	Real-time hand gesture recognition based on deep learning YOLOv3 Model	2021
4	Deep residual learning for image recognition	2016
5	MediaPipe hands: On-device real-time hand tracking	2020

DATASET USED:

- **Perceptron model:** In order to train the perceptron model we have used our own dataset consisting of 1000 thresholded images of a closed fist and an open hand showing all 5 fingers each and to validate the model we have used a total of 1000 images(500 images per label) of both closed fist and an open hand showing 5 fingers.
- **CNN model:** In order to train the CNN model we have used our own dataset of a total of 7999 images(1000 images per label) of 8 separate hand gestures consisting of fist, five, okay, peace,rad, straight, thumbs and no hands. To validate this model we have used 4000 images(500 images per label) of the same gestures as mentioned.
- **Nvidia GestureNet:** It was trained on a proprietary dataset with more than 150K images. The training dataset contains hand crops from users facing cameras at various heights with variations in user, backgrounds and illumination and it has been validated against 10000 proprietary images across a variety of environments, backgrounds and illumination.

- **MediaPipe:** To obtain ground truth data, mediapipe uses the following datasets addressing different aspects of the problem:
    1. In-the-wild dataset: This dataset contains 6K images of large variety, e.g. geographical diversity, various lighting conditions and hand appearance. The limitation of this dataset is that it doesn't contain complex articulation of hands.
    2. In-house collected gesture dataset: This dataset contains 10K images that cover various angles of all physically possible hand gestures. The limitation of this dataset is that it's collected from only 30 people with limited variation in background. The in-the-wild and in-house dataset are great complements to each other to improve robustness.
    3. Synthetic dataset: To even better cover the possible hand poses and provide additional supervision for depth, it renders a high-quality synthetic hand model over various backgrounds and map it to the corresponding 3D coordinates. It use a commercial 3D hand model that is rigged with 24 bones and includes 36 blendshapes, which control fingers and palm thickness. The model also provides 5 textures with different skin tones. WIt created video sequences of transformation between hand poses and sampled 100K images from the videos. It rendered each pose with a random high-dynamic-range lighting environment and three different cameras

  For the palm detector, it only uses in-the-wild dataset, which is sufficient for localizing hands and offers the highest variety in appearance. However, all datasets are used for training the hand landmark model. It annotate the real-world images with 21 landmarks and uses projected ground truth 3D joints for synthetic images. For hand presence, it selects a subset of real-world images as positive examples and samples on the region excluding annotated hand regions as negative examples. For handedness, it annotates a subset of real-world images with handedness to provide such data

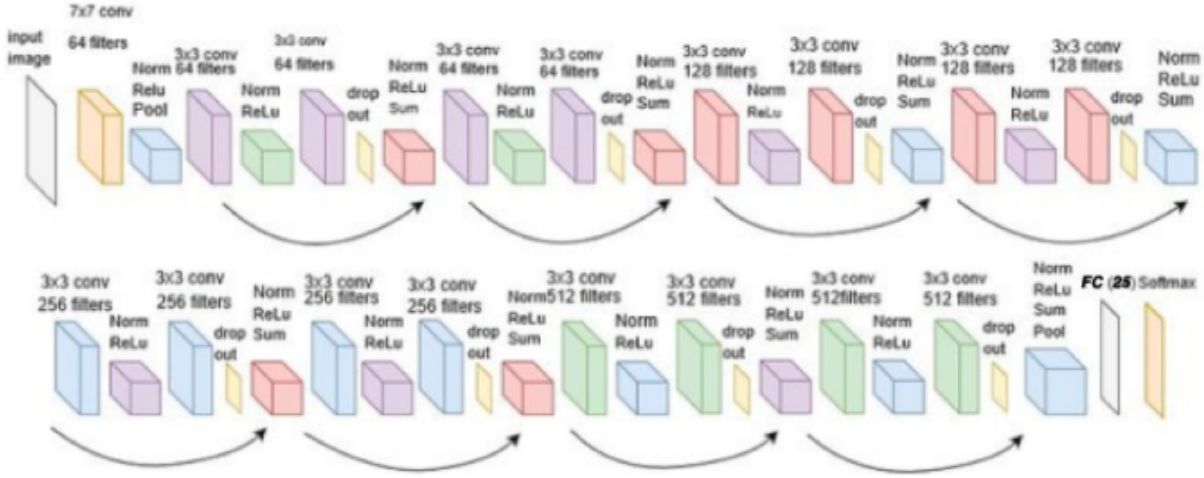
PROPOSED ARCHITECTURE

For Perceptron Model: The network contains two layers(Dense) one input and one output with weights. The activation function used is 'relu and softmax'. The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution. The loss function, also called the objective function, is the evaluation of the model used by the optimizer to navigate the weight space. The optimizer used is 'adam'. Adam optimization is a stochastic gradient descent method that is based on adaptive estimation of first-order and second-order moments.

For CNN Model: The network contains a total of 10 layers out of which 4 are of Convolution2D, 2 are of MaxPooling2D, 2 are of Dense, 1 is of Flatten and the last one is of Dropout.

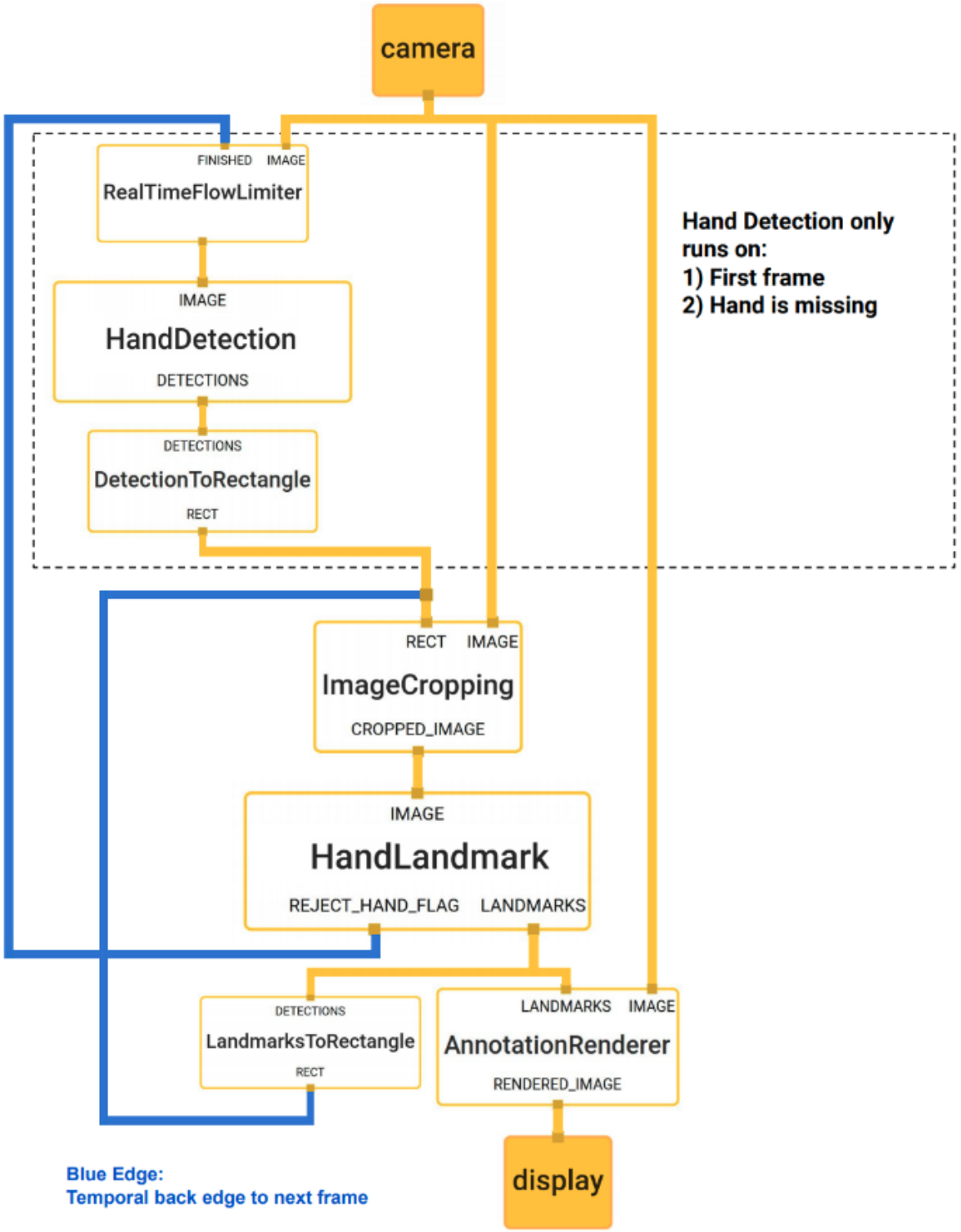
We have used ‘relu’ as the activation function. ReLU stands for Rectified Linear Unit, and is the most commonly used activation function in neural networks. ReLU activation function ranges from 0 to infinity, with 0 for values less than or equal to 0, that is, 0 for negative values. The RMSprop optimizer is similar to the gradient descent algorithm with momentum. The RMSprop optimizer restricts the oscillations in the vertical direction. Therefore, we can increase our learning rate and our algorithm could take larger steps in the horizontal direction converging faster.

For GestureNet: It is based on the ResNet-18 convolutional neural network architecture. ResNet-18 uses a 18 layer plane net in which the convolutional layers mostly have 3×3 filters and follows two simple design rules: (i) for the same output feature map size, the layers have the same number of filters; and (ii) if the feature map size is halved, the number of filters is doubled so as to preserve the time complexity per layer. But it also adds a shortcut connection to each pair of 3×3 filters.



Architecture of ResNet-18 model

For MediaPipe: First it trains a palm detector since estimating bounding boxes of rigid objects like palms and fists is significantly simpler than detecting hands with articulated fingers. In addition, as palms are smaller objects, the non-maximum suppression algorithm works well even for the two-hand self-occlusion cases, like handshakes. Moreover, palms can be modelled using only square bounding boxes, ignoring other aspect ratios, and therefore reducing the number of anchors by a factor of 3~5. Second, it use an encoder-decoder feature extractor similar to FPN for a larger scene-context awareness even for small objects. Lastly, it minimize the focal loss during training to support a large amount of anchors resulting from the high scale variance. High-level palm detector architecture is shown below.

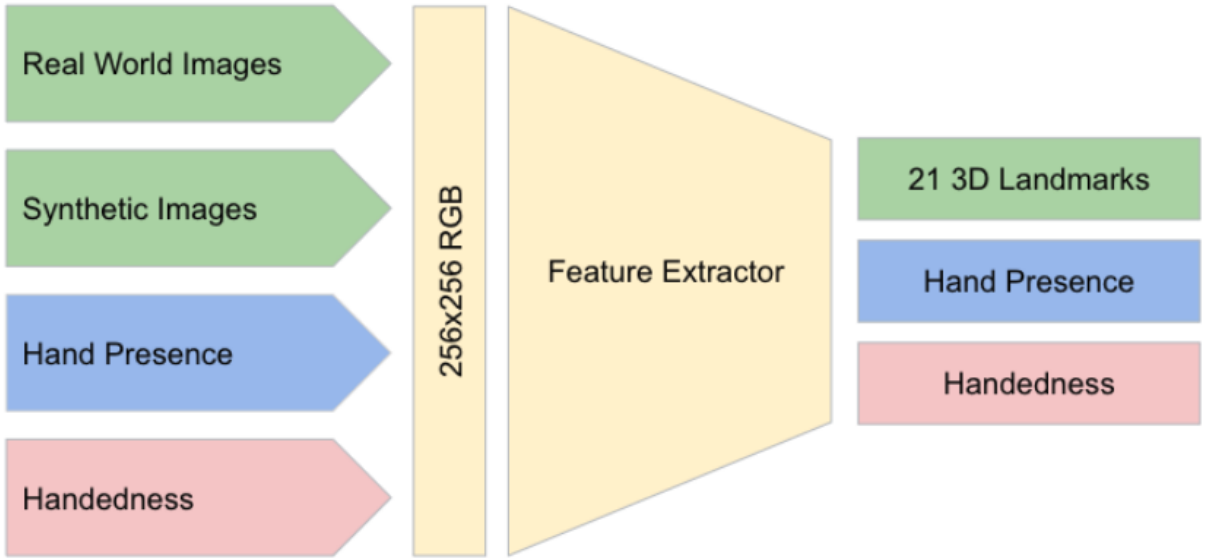


After running palm detection over the whole image, its subsequent hand landmark model performs precise landmark localization of 21 2.5D coordinates inside the detected hand regions via regression. The model learns a consistent internal hand pose representation and is robust even to partially visible hands and self-occlusions. The model has three outputs:

1. 21 hand landmarks consisting of x, y, and relative depth.

2. A hand flag indicating the probability of hand presence in the input image.

3. A binary classification of handedness, e.g. left or right hand.



It use the same topology as [9] for the 21 landmarks.The 2D coordinates are learned from both real-world images as well as synthetic datasets as discussed below, with the relative depth w.r.t. the wrist point being learned only from synthetic images. To recover from tracking failure, it develops another output of the model similar to [10] for producing the probability of the event that a reasonably aligned hand is indeed present in the provided crop. If the score is lower than a threshold then the detector is triggered to reset tracking. Handedness is another important attribute for effective interaction using hands in AR/VR. This is especially useful for some applications where each hand is associated with a unique functionality. Thus it develops a binary classification head to predict whether the input hand is the left or right hand. Its setup targets real-time mobile GPU inference, but it has also designed lighter and heavier versions of the model to address CPU inference on the mobile devices lacking proper GPU support and higher accuracy requirements of accuracy to run on desktop, respectively.

RESULT AND EXPERIMENTAL ANALYSIS

Our first initial model which used a perceptron had a validation accuracy of 96.6% which does sound incredible however the model is only capable of detecting just 2 hand gestures, namely a fast and a five finger palm.

Bound by the limitations of a perceptron model we had to shift to a simple CNN in order to increase our scope out of just 2 gestures to a total of 8 gestures. That model gave us a validation accuracy of 83.85% which is actually quite good but it had its own limitations. The model still needed to work on a still background and had to be calibrated first and if by any chance the camera moved even a little the whole model got rendered as completely useless. Due to the shortcomings of this model we decided to increase our scope further to adopt the hand gesture detection pipelines available.

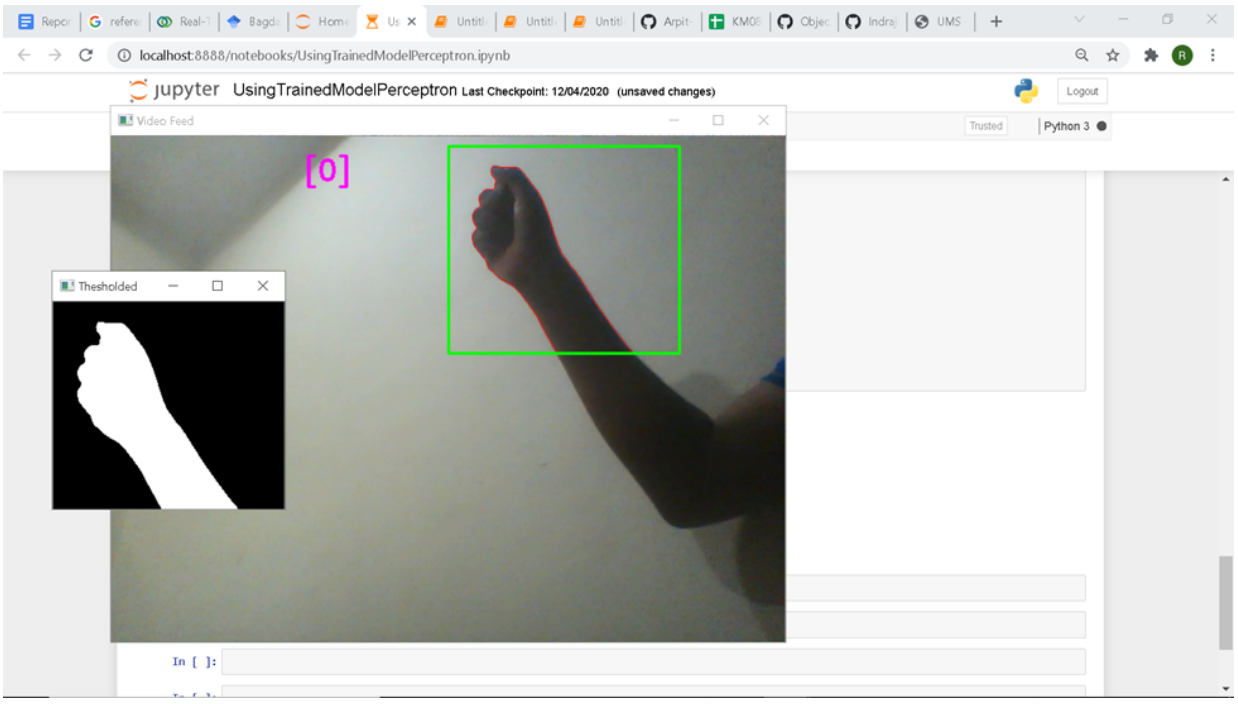
One of the first pipeline we stumbled upon was Nvidia GestureNet which we built using the TAO toolkit. It was trained on a proprietary dataset but had an incredible accuracy. The Key Performance Indicators of it on its own evaluation data set is as follows:

Model		**GestureNet**	
Content	Precision	Recall	F1-Score
Evaluation set	85%	85%	85%

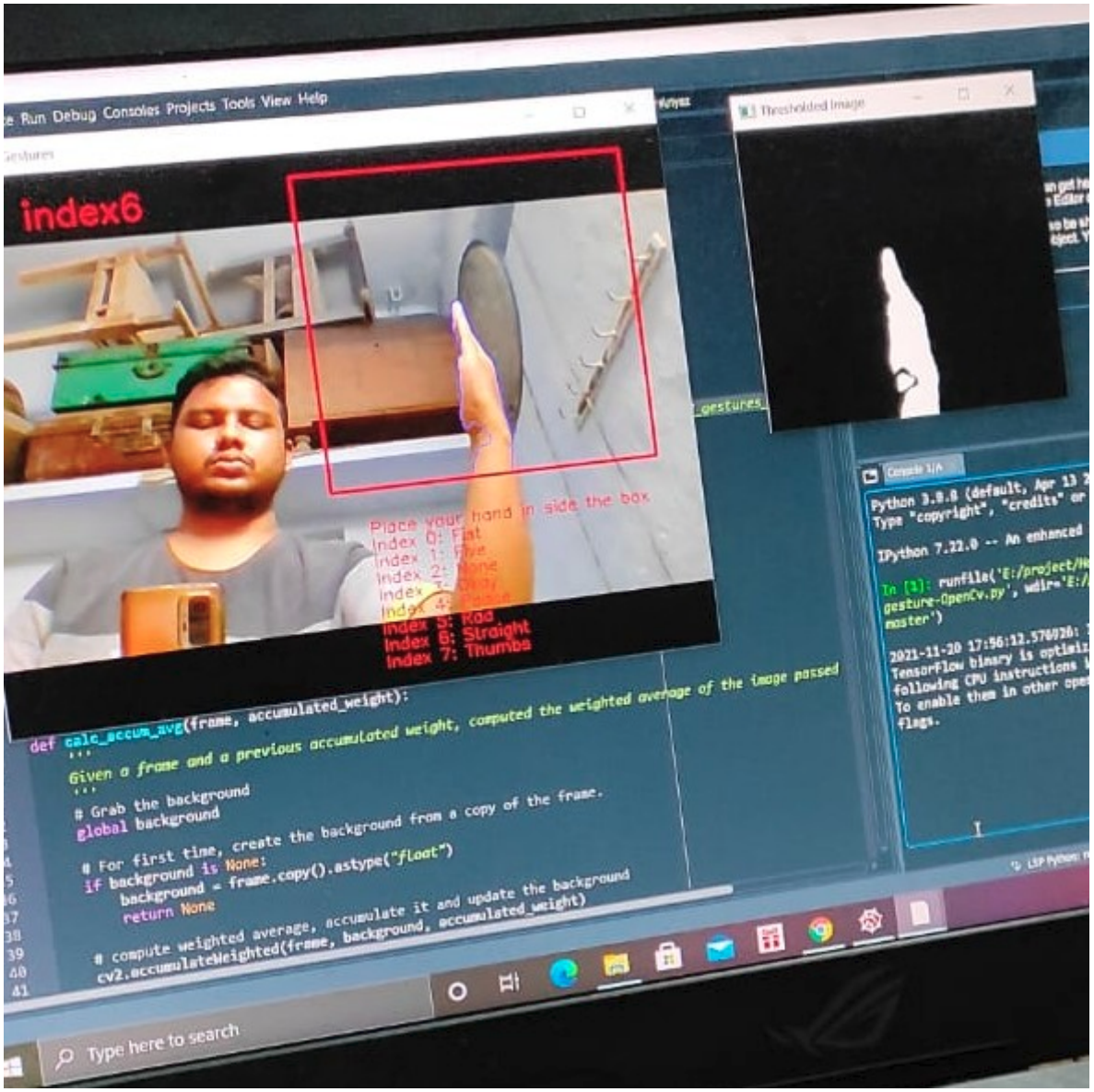
It also didn't need the stabilized background which our previous model did, however it was still not suitable for our project because it still didn't have the modular approach that we needed. It could only perform on the gestures it was initially trained with and adding another gesture needed huge amounts of data to train the gesture which we lacked.

In order to get a more modular approach to our project we had to shift to MediaPipe hands, not only did this model had incredible accuracy considering that it created 21 landmarks across each palms and had that modularity to it since we could identify gestures based on how those landmarks interact with each other but it also had great performance metrics when it came to speed.

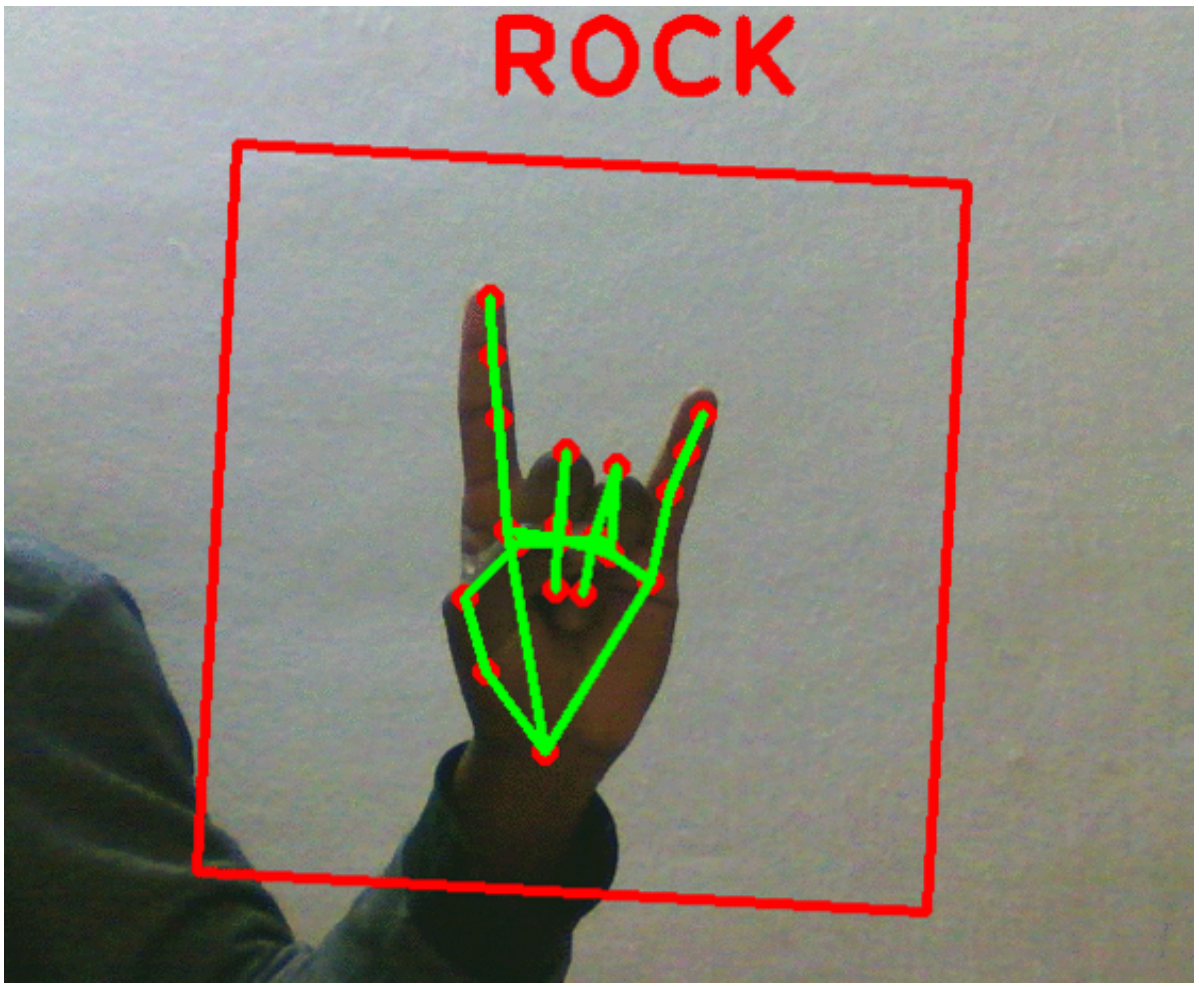
SCREENSHOTS



Hand gesture detection using perceptron



Hand gesture detection using CNN



Hand gesture detection using MediaPipe

CONCLUSION AND FUTURE SCOPE

In the end MediaPipe hands were the most versatile and the most suitable pipeline for our use case. The hand detection model is already quite refined and works like a charm on python. However we can still improve it further by building the project in C++ using bazel and implementing our own hand landmark detection calculators to identify various gestures. Not only that would increase the performance of the model (in terms of speed and FPS) but it would also allow us to create separate applications based on the same calculator as long as it is required to detect the same gesture.

REFERENCES

1. J. L. Raheja, A. Chaudhary, and K. Singal, “Tracking of fingertips and centers of palm using KINECT,” in Proceedings of the 2nd International Conference on Computational Intelligence, Modelling and Simulation (CIMSim '11), pp. 248–252, September 2011.
2. View at: Publisher Site | Google Scholar

3. Y. Wang, C. Yang, X. Wu, S. Xu, and H. Li, “Kinect based dynamic hand gesture recognition algorithm research,” in Proceedings of the 4th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC '12), pp. 274–279, August 2012.
4. View at: Publisher Site | Google Scholar
5. M. Panwar, “Hand gesture recognition based on shape parameters,” in Proceedings of the International Conference on Computing, Communication and Applications (ICCCA '12), pp. 1–6, February 2012.
6. View at: Publisher Site | Google Scholar
7. Z. Y. Meng, J.-S. Pan, K.-K. Tseng, and W. Zheng, “Dominant points based hand finger counting for recognition under skin color extraction in hand gesture control system,” in Proceedings of the 6th International Conference on Genetic and Evolutionary Computing (ICGEC '12), pp. 364–367, August 2012.
8. Nvidia GPU Cloud Catalog | GestureNet
9. Tomas Simon, Hanbyul Joo, Iain A. Matthews, and Yaser Sheikh. Hand keypoint detection in single images using multiview bootstrapping. CoRR, abs/1704.07809, 2017.
10. Yury Kartynnik, Artsiom Ablavatski, Ivan Grishchenko, and Matthias Grundmann. Real-time facial surface geometry from monocular video on mobile gpus. CoRR, abs/1907.06724, 2019
11. Nvidia Docs | Tao toolkit
12. MediaPipe | MediaPipe Hands

GITHUB LINK

https://github.com/Arpit-Banerjie125/INT248