Hand Gesture Recognition

Navateja Cherukuri
ncpfg@umsystem.edu

Durga Vara Prasad Reddy Maryada
dmpqq@umsystem.edu

Navya Lakshmi Kusam
nlkd7d@umsystem.edu

Sai Venkata Anudeep Sanchula
asgzq@umsystem.edu

ABSTRACT
In this project, we have developed a hand gesture recognition model that predicts the gestures that are captured using the camera.

INTRODUCTION
Computers and computer devices are becoming an ever-increasing part of our daily lives. Because of the increasing need for such computing devices, the development of simple and effective computer interfaces was required. As a consequence of the vast application possibilities in human machine interface, systems based on vision-based interaction and control are becoming more common, and as a result, gesture recognition is becoming increasingly popular in the research community. Any vision-based interface is more convenient, practical, and natural than a mouse and keyboard because of the intuitiveness of gestures. Gesture recognition can be accomplished using one of three methods:

(i) glove-based wearable devices

(ii) three-dimensional hand key point locations

(iii) raw visual data.

The first way needs the wearer to wear an additional device with many wires, but it produces good results in terms of accuracy and speed.

The second, on the other hand, requires an extra step of hand-keypoint extraction, which adds time and computational cost.

Finally, all that is necessary is an image capturing sensor that is independent of the user, such as a camera, infrared sensor, or depth sensor. This technique is the most practical since the user does not need to wear a heavy device in order to acquire enough identification accuracy and computing speed. It is vital for the infrastructure of any gesture recognition system to be practical. After all, we want to use it in real-world situations.

PROBLEM STATEMENT
With the rise of ubiquitous computing, traditional user interface methods such as keyboard, mouse, and pen are no longer adequate. Because of the limitations of these devices, the command set that may be used is similarly restricted. The direct use of hands as an input device can be employed to provide natural interaction.

RELATED WORK
CNNs have been more popular as a general-purpose computer vision tool due to their ability in object recognition and categorization. First, CNNs were used for video action and activity identification, and they have reached state-of-the-art performance in video analysis applications. CNNs have been used to extract spatio-temporal information from video footage in a variety of ways. Due to its performance in static pictures, 2D CNNs were first used in video analysis based techniques. There are two-dimensional CNNs that use video frames as multi-channel input.

The Temporal Segment Network (TSN) uses 2D CNNs to separate video into segments, then employs spatio-temporal modeling to extract information from color and optical flow modalities for action identification. Video frames are first segmented into their individual characteristics using a 2D convolutional neural network (CNN), and then the LSTM algorithm is used to represent the overall temporal structure of the video. Because there are several viable 2D CNN designs that can be pretrained using the Kaggle dataset, the strength of all these techniques is derived.

Two-dimensional convolutional neural networks (2D CNNs) perform well on video analysis tasks, however they are restricted in their ability to understand motion patterns and temporal data. 3D CNNs that employ 3D convolutions and pooling to collect discriminative information along both spatial and temporal dimensions are the best choice for this task. 3D CNNs are different from 2D CNNs in that they use video frames as inputs instead of only one. Additionally, we use 3D CNN variations.

SOLUTION
To give a working solution, we created a gesture detection system that uses raw video data and deep convolutional neural networks (CNNs). For video-based tasks like activity detection, action localization, and gesture recognition, as well as image-based ones like object identification, picture segmentation, and classification, CNNs now give cutting-edge results. In real-time gesture recognition applications, the system must meet specific requirements, such as (i) relatable accuracy for the classification, (ii) fast response time, (iii) Efficiency of resources, and (iv) One-time activation for every gesture that is performed . All these elements are required for an effective

real-time vision-based gesture recognition systems. The majority of past research, on the other hand, has focused primarily on enhancing gesture recognition classification accuracy while ignoring the remaining components.

DATASET

In this project, we have used hand gestures data from kaggle from the following sources.

https://www.kaggle.com/sarjit07/ hand-gesture-recog-dataset

https://www.kaggle.com/muhammadkhalid/ sign-language-for-numbers

https://www.kaggle.com/ahmedkhanak1995/ sign-language-gesture-images-dataset

From the above datasets, we have obtained the hand gesture training and test images and the corresponding classifications for 1,2,3,4,5,6,7,8,9 (Numbers) ,C Shape, Call me, Down, Fingers Crossed, First, Index, L, Moved Fist, Moved palm, Ok Super, Palm, Paper palm, Peace, Rock fist, Scissor, Thumb, Up, YO-YO hand gestures.

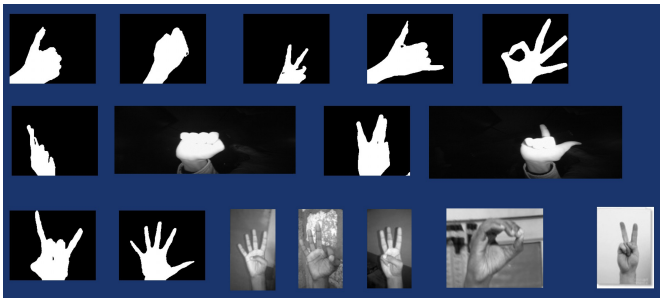

Figure 1. Sample Gestures

IMPLEMENTATION

Pre-Processing

In the Pre-Processing stage, we have resized the images in the dataset to 120*120 size. Next we have Defined the characters the key of the category dictionary is the folder name, and value is an integer. Then from each category we have taken 150 images for training and 50 images for testing. and then shuffled the training data using random.shuffle function. then we have Scaled the image pixel values between 0 and 1 for both training and testing data.

Model

After trying out different layers at different layers, we have converged at a model with the configuration as below.

and we have trained the model with above configuration using the training data obtained in the pre-processing stage. and saved the model to a file model.h5, this file will be used in the prediction stage to load the model and predict the gesture of a new image.[1]

Once we have trained the model with the training data, we have evaluated our model using the test data and the model has an accuracy of 76

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 58, 58, 32)        832

max_pooling2d (MaxPooling2D  (None, 29, 29, 32)        0
)

conv2d_1 (Conv2D)            (None, 27, 27, 64)        18496

max_pooling2d_1 (MaxPooling  (None, 13, 13, 64)        0
2D)

conv2d_2 (Conv2D)            (None, 11, 11, 64)        36928

max_pooling2d_2 (MaxPooling  (None, 5, 5, 64)          0
2D)

flatten (Flatten)            (None, 1600)              0

dense (Dense)                (None, 128)               204928

dense_1 (Dense)              (None, 29)                3741

=================================================================
```

Figure 2. Model Summary

Prediction/Application

We have built a python-flask application, which when called using an API, will call a method, where we read the image from the camera using open-CV library. and since for gesture recognition, we don't need the colour of the image, we convert it into gray scale. and followed the below steps in order to obtain the prediction of the gesture

Identifying the Background

In order to identify the hand in the image captured using the camera. we read the background image only for the first 30 frames and then take the average of those images, this average serves as reference background to distinguish the hand in the image.

Identifying the hand

Once we have the reference background image data, for the subsequent frames we deduct the reference background image from the input image obtained from camera, converted to gray scale. Once the background is deducted, we obtain the hand image in the input image.

Prediction

Once the hand is identified in the above step, we use this hand image captured and scale the image to that of the training, testing data sets that is 120*120, and use this image as input to model.predict function for predicting the gesture. This process of prediction is done at every 100th frame which is the window size we have taken, as the input is a streaming data.

FUTURE WORK

Now that we are able to predict the hand gesture given by the end user using our application, we can extend the functionality so as to allow the end user to configure specific tasks for specific gestures and then enable the application to perform those tasks like calling a phone number on call me gesture etc..

REFERENCES

https://machinelearningmastery.com/what-is-deep-learning/

https://brohrer.github.io/how_convolutional_neural_networks_work.html

https://keras.io/guides/sequential_model/

https://keras.io/api/models/

https://www.kdnuggets.com/2018/04/top-16-open-source-

deep-learning-libraries.html
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
http://jalammar.github.io/visual-interactive-guide-basics-neural-networks/
https://www.jeremyjordan.me/nn-learning-rate/
https://keras.io/api/losses/probabilistic_losses/
https://keras.io/activations/
https://keras.io/optimizers/
https://keras.io/losses/
https://keras.io/initializers/
https://keras.io/callbacks/
https://keras.io/metrics/
https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/
https://pylessons.com/CNN-tutorial-introduction/