Accomplishing Common Tasks Using Gestures

Saurabh Tambolkar
State University of New York at Buffalo
50412968
stambolk@buffalo.edu

Shantanu Kumar
State University of New York at Buffalo
50418500
skumar39@buffalo.edu

Abstract

A hand gesture based interface provides much higher flexibility while also being user friendly because the user has the freedom of operating a machine using only his hand in front of his/her camera. Countless researchers are drawn to the flexibility and efficiency of these non-contact communication methods and have made them consider employing them to support human computer interaction. A gesture can be defined as the motion of the body that intends to communicate with other agents in the environment. To be successful at communicating via gestures, the sender and receiver must have mutual agreement on what constitutes as a gesture first, otherwise there is a large scope of overall ambiguity. Gesture recognition can be defined as the ability of a machine, generally a computer to understand human gestures and perform certain commands based on those gestures. The main goal here is to develop a system that can identify and understand specific gestures and use them in a system to increase productivity and ease of use. Since a camera is available in majority of devices these days, we can employ it to capture gestures and then map those gestures to specific functions to accomplish common tasks. In this project, we employ static gestures and aim to use convolutional neural networks to recognize hand gestures and accomplish basic tasks that will make the everyday life of users easier. Here we taken the example of two simple applications to demonstrate our point, the possibilities are practically limitless. In the first application, we have utilized gestures to control the volume of music playing in the background. We tend to listen to music fairly frequently while doing everyday tasks as a means of leisure, thus, being able to control it via gestures without reaching out to interact with physical or touch controls is very helpful. In the second application, we have taken up something a little more complex. We have employed 5 different gestures to achieve the functionality of the cursor, 4 gestures accounting for movement in the horizontal and vertical direction and 1 gesture to account for the mouse click. The cursor is just a simple example and be replaced by anything that moves in a plane, for example, a remote controlled car. It goes without saying that as the number of recognized gestures increases, the scope of tasks that can be accomplished increases exponentially. It should be noted that it is the duty of the designer to come up with gestures that are intuitive enough for the user to remember. Having complex gestures for tasks that require a high degree of coordination and rapid decision making will only hinder the user experience.

1 Introduction

In the recent years, the most predominant form of communication between the machine and the user is direct contact. This communication channel is based on devices such as a remote control, mouse,

keyboard, touchscreen and other direct contact methods. In the growing field of computer vision, gesture recognition is also one of the most important and growing problems and has been in many fields, such as human computer interaction, virtual reality systems, and sign language recognition. A hand gesture based interface provides much higher flexibility while also being user friendly because the user has the freedom of operating a machine using only his hand in front of his/her camera. Several applications that employ static hand gesture systems are sign language processors, automated television control devices, smart home interaction controls, controlling a software interface, playing games using consoles (Nintendo Wii) and controlling virtual environments.

Gestures may be classified as dynamic or static. A dynamic gesture generally varies over a period of time meanwhile static gestures, as the name suggests, tend to remain almost unchanged over time. Static gesture recognition methods have significant shortcomings and limitations. They can only recognize a single shape of the hand but struggle considerably with its spatial and state variations. As an example, it can recognize whether the hand is in a state of "unfolded" or "held" state, but it cannot interpret if the state changes from one state to the other. Dynamic gesture recognition considers the spatial and temporal information of the whole procedure and recognizes the process of change of the target object, which in turn has serious research implications. The main goal here is to develop a system that can identify and understand specific gestures and use them in a system to increase productivity and ease of use. In this project, we employ static gestures.

Historically, wearable data gloves were regularly used to capture the angles and positions of each joint in the user's gesture. The difficulty and cost of a wearable sensor have restricted the widespread use of such a method. Advances in computer vision technology have allowed the use of biological characteristics of human beings and brought about a shift in human computer interaction from traditional ways to new methods. Our hands are the most flexible part of our body and thus, hand gestures can express rich and multiple forms of communication and thus, are widely used for communication between humans and devices smartphones, automobile infotainment systems, robotics. Gestures will replace touch and wire controlled input devices. Different sensors have different sensing capabilities. Mostly, a single sensor is used for gesture based interaction. In order to recognize a gesture, raw data needs to be collected by the sensors. There can be various ways to getting this data, for example, using a contactless sensor such as radar for hand movement detection or a wearable sensor such as a glove, which can measure the pressure applied by the fingers around the wrist.

Deep Learning (DL) is an emerging field and is a sub category of machine learning inspired by the function of the human brain and its structure. It employs multiple hidden neural network layers for better learning of a model. The Convolutional-Neural-Network (CNN) is a very famous DL model which is used in image-based applications. Nowadays, DL is used in visual object detection and recognition, speech recognition and many other applications. Multi-column deep CNNs that employ multiple parallel networks have been shown to improve recognition rates of single networks by 30-80 percent for various image classification task. In recent years, many hand gesture recognition methods have been proposed using deep learning techniques. Image enhancement techniques are applied to increase the quality of input images. To apply deep learning based gesture recognition, the dataset needs to be large; therefore to enrich the input dataset, data augmentation is employed in this work using scaling, translating, rotating and shearing techniques. Dynamic gesture recognition falls under the category of video classification since the dataset is mostly available in the form of video frames. Hence, both spatial and temporal domains features are used. Dynamic gesture recognition is a difficult task because the images obtained from video recordings does not have consistent pixels; the camera is not fixed at one position and every person performs the same gesture in a different way. Gesture includes different background with hand and arm continuous movement due to which it is not easy for an algorithm to predict the gestures with very high accurately.

2 Related Work

There has been a lot in interest in hand gesture recognition in the recent years to make quality of life improvement in various fields. Some the the notable work has been shown below.

In Hand Gesture Recognition with 3D Convolutional Neural Networks, the authors introduce a hand gesture recognition system that utilizes depth and intensity channels with 3D convolutional neural networks. Motivated by Molchanov et al, they interleave the two channels to build normalized spatio-temporal volumes, and train two separate sub-networks with these volumes. To reduce potential over-fitting and improve generalization of the gesture classifier, the authors proposed an effective spatio-temporal data augmentation method to deform the input volumes of hand gestures. The augmentation method also incorporates existing spatial augmentation techniques. This work bears similarities to the multi-sensor approach of Molchanov et al., but differs in the the use of two separate sub-networks and data augmentation. The authors demonstrate that their system, with two sub-networks, that employs spatio-temporal data augmentation for training, outperforms both a single CNN and the baseline feature-based algorithm on the VIVA challenge's dataset. They evaluated the performance of our dynamic hand gesture recognition system using leave-one-subject-out cross-validation on the VIVA challenge's dataset. They used data from one of the 8 subjects for testing and trained the classifier with data from the 7 remaining subjects; then repeated this process for each of the 8 subjects and averaged the accuracy. Furthermore, the final classifier that combined the outputs of LRN and HRN outperformed the baseline method by 13.0. Moreover, 52% of the final classifier's errors were associated with the second most probable class. The results indicate that our CNN-based classifier for in-car dynamic hand gesture recognition considerably outperforms approaches that employ hand-crafted features.

In Dynamic Hand Gesture Recognition Using 3D-CNN and LSTM Networks, a deep learning-based model which is combination of 3D-CNN and LSTM is proposed for recognizing dynamic hand gestures. To evaluate the proposed technique, the 20BN-jester dataset is used. The 20BN-jester consists of 148,092 labeled video clips showing different people performing different dynamic hand gestures. The dataset has approximately 5000 video clips per class which are separated into training, validation and test sets. Due to computational restrictions, only 15 classes have been used in this paper. As discussed in the later sections of the paper, the proposed model attained an accuracy of 97% on unseen data taken from the test set. Deep learning models need more data for improved training and subsequent performance. To achieve this, data augmentation techniques are used to modify the current dataset and create more variations of the images which will improve the model learning. The features obtained from the 3D-CNN layers passes to the L2 batch normalization layer and are then fed to the LSTM layer. For the MobileNet-V2 + LSTM model, pre-trained weights of the ImageNet dataset were used which gave a validation accuracy of 84% at 20 epochs. The results however did not improve further with validation loss not going below 0.25. The accuracy was reasonable but the real-time gestures prediction through a webcam was not accurate. After this, L2 batch normalization was introduced to MobilNet-V2+LSTM model and the accuracy improved to 87%, which was better but not acceptable as compared to other techniques proposed in the literature. A light-weighted model consisting of 3D-CNN+LSTM with L2-batch normalization was used which had 3.7 million training parameters.

Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation emphasizes on recognition of static hand gestures by building a model using CNN that can analyze large amount of image data and recognize static hand gestures. The other objectives are to investigate the existing methods of gesture recognition and analyzing the effect of data augmentation in deep learning. The authors selected 10 static gestures (Index, Peace, Three, Palm, Opened, Palm Closed, OK, Thumbs, Fist, Swing, Smile) to recognize. Each class has 800 images for training and 160 images for testing purpose. So total number of images is 8000 or training and 1600 for testing. The experimental result shows that the model which was augmented with temporary data achieved 97.12% accuracy which is about 4% higher than the model without any augmented data. The accuracy of augmented model was higher than non-augmented model at each epoch. It also shows that the progress of accuracy on augmented model was faster than non-augmented model. The loss of

augmented model was also less than non-augmented model. The same dataset was passed as input to Support Vector Machine (SVM) and K Nearest Neighbors (KNN) models. These models achieved accuracy of 72% and 75% respectively. One of the reasons of poor performance showed by SVM and KNN is the adaptability issue with non-linear dataset. Since, the dataset was provided in a raw format, their accuracy were lower than CNN.

In Dynamic gesture recognition based on 2D convolutional neural network and feature fusion, to improve the problem of large network model parameters and training difficulties, the authors propose a strategy based on dual-channel 2D CNN and feature fusion. First, the optical flow frames of the video data were extracted using the fractional Horn and Schunck (HS) optical flow method, and then five original key frames and optical flow key frames were extracted separately using an improved clustering algorithm and subjected to a horizontal stitching operation. The stitched original keyframe feature map is used to represent the spatial features in the video data, and the optical flow keyframe stitching map represents the temporal features in the video data. This method not only preserves the spatial and temporal features of the video, but it also greatly reduces the size of the dataset and improves the training efficiency. Most current algorithms on dynamic gesture recognition using 2D CNN serialize the video datasets as a chart or a single image, which loses the information on the variation of key spatio-temporal features. The accuracy of the proposed method is 97.6% on the Northwestern University datasets and 98.6% on the Cambridge datasets. The time of the proposed algorithm is 9.93 s on the Northwestern University gesture dataset and 4.02 s on the Cambridge gesture dataset, both of which are more significant improvements over previous algorithms.

In Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network, a score-level fusion technique between two fine-tuned CNNs such as AlexNet and VGG-16 is proposed. An end-to-end fine-tuning of the deep CNNs such as AlexNet and VGG-16 is performed on the training gesture samples of the target dataset. Then, the score-level fusion technique is applied between the output scores of the fine-tuned deep CNNs. The performance of recognition accuracy is evaluated on two publicly available benchmark American Sign Language (ASL) large-gesture class datasets. A real-time gesture recognition system is developed using the proposed technique and tested in subject-independent mode. The results show that the proposed technique performs better in terms of mean accuracy (average ± standard deviation) compared to both fine-tuned CNNs on the above datasets. The reason for this is described below. The LOO CV technique is a user-independent CV technique. In this technique, the performance of the trained model is evaluated using the gesture samples of the user, who does not take part in the model development. However, in regular CV, the gesture samples of all the users in the dataset take part in the training and testing processes. Hence, this CV test is user-biased. Therefore, the model performance using the regular CV test is higher than the LOO CV test.

3 Dataset

Our dataset is fairly simple. Since we are using hand gestures, our dataset consists of hand images portraying different gestures. These are captured using a webcam and then provided to our convolutional neural network as an input. The various gestures used in our project are shown below.

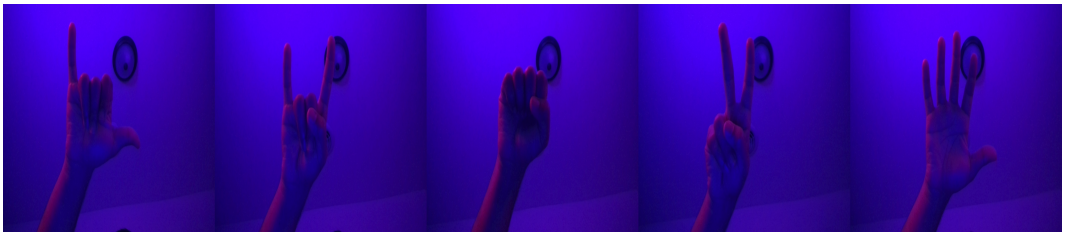

Figure 1: Gestures

We have utilized the camera provided in our PCs to create our dataset. We used the images of our hands to train the model and tested it using the hand of others to account for the change in anatomy. We also captured images in varying environment lighting to account for real world usage.

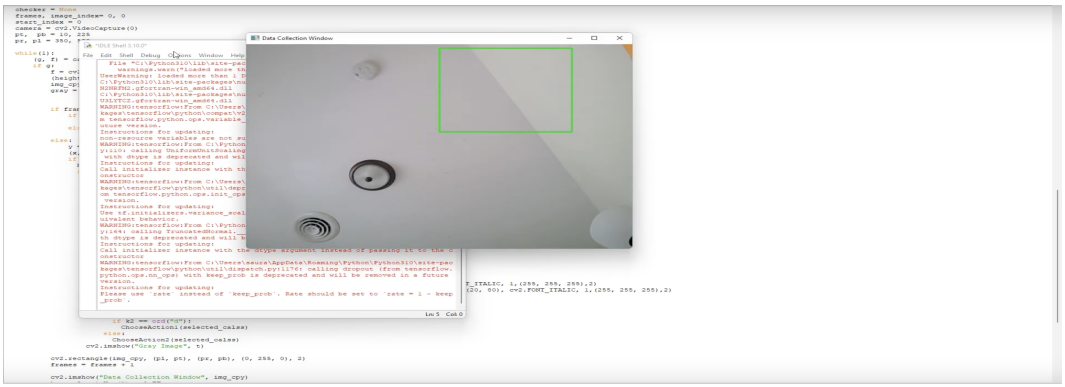

Figure 2: Capturing the Dataset

In the image shown above, we capture images when they are placed inside the box shown. We move our hand inside the box, press the key that prompts the script to capture images and it clicks 100 images of our gesture.

When we run the dataset generation script once, it takes 100 images of a particular gesture using the running average by making use of OpenCV. We have trained each gesture using a total of 1000 images. We train a total of 5 gestures. Effectively, we used 5,000 images to train our model. It goes without saying that the model can be extended to more than 5 gestures.

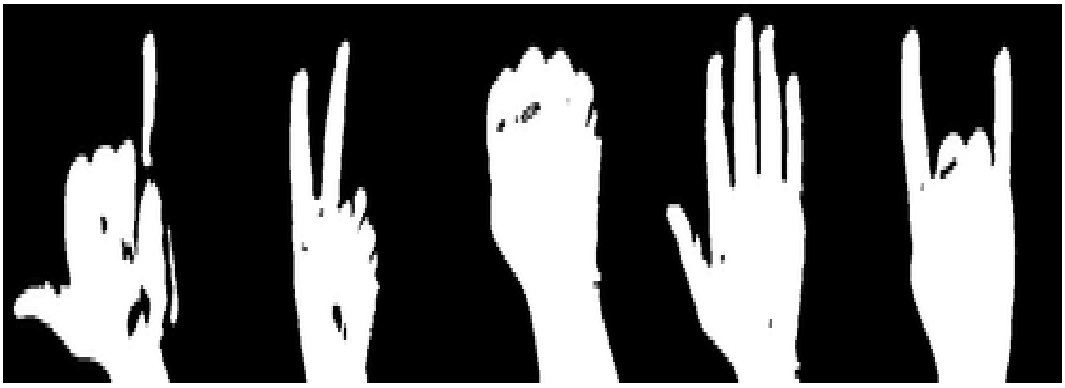

Figure 3: Gestures in Grayscale

We also took the images for our testing dataset from varying distances to the camera. We did this since in real world settings, we do not expect the user to make a gesture right in front of the camera. An ideal neural network should be able to work at distances as well.

4 Proposed Method

To complete this project, the following must be installed in your PC.

- Python 3
- Tensorflow
- OpenCV
- TFLearn

- NumPy
- PIL Pillow
- imutils

The first step in hand gesture recognition is obviously to find the hand region by eliminating all the other unwanted portions in the video sequence. A video sequence is just a collection of frames or collection of images that runs with respect to time.

First, we need an efficient method to separate foreground from background. To do this, we use the concept of running averages. We make our system to look over a particular scene for 30 frames. During this period, we compute the running average over the current frame and the previous frames. By doing this, we essentially tell our system that the running average of those 30 frames is the background. After figuring out the background, we bring in our hand and make the system understand that our hand is a new entry into the background, which means it becomes the foreground object. We accomplish this using the technique of background subtraction. After figuring out the background model using running averages, we use the current frame which holds the foreground object (hand in our case) in addition to the background. We calculate the absolute difference between the background model which is updated over time and the current frame, which has our hand to obtain a difference image that holds the newly added foreground object

To detect the hand region from this difference image, we need to threshold the difference image, so that only our hand region becomes visible and all the other unwanted regions are painted as black. Thresholding is the assignment of pixel intensities to 0's and 1's based a particular threshold level so that our object of interest alone is captured from an image.

After thresholding the difference image, we find contours in the resulting image. The contour with the largest area is assumed to be our hand. Contour is the outline or boundary of an object located in an image.

Running average is calculated using the formula given below.

$$dst(x, y) = (1 - a).dst(x, y) + a.src(x, y) \tag{1}$$

where, src(x,y) is the source image or input image (1 or 3 channel, 8-bit or 32-bit floating point)

dst(x,y) is the destination image or output image (same channel as source image, 32-bit or 64-bit floating point)

a is the weight of the source image (input image)

Next, we threshold the difference image to reveal only the hand region. Finally, we perform contour extraction over the thresholded image and take the contour with the largest area (which is our hand). We return the thresholded image as well as the segmented image as a tuple. The math behind thresholding is pretty simple. If x(n) represents the pixel intensity of an input image at a particular pixel coordinate, then threshold decides how nicely we are going to segment/threshold the image into a binary image.

$$x(n) = 1 \; if \; n >= threshold, otherwise \; 0 \tag{2}$$

As shown earlier in the running average equation, this threshold means that if you set a lower value for this variable, running average will be performed over larger amount of previous frames and vice-versa. Instead of recognizing gestures from the overall video sequence, we will try to minimize the recognizing zone (or the area), where the system has to look for hand region.

Next, we take out only the region of interest (i.e the recognizing zone), using simple NumPy slicing. We then convert this ROI into grayscale image and use gaussian blur to minimize the high frequency components in the image. Until we get past 30 frames, we keep on adding the input frame to our

running average function and update our background model. Please note that, during this step, it is mandatory to keep your camera without any motion. Or else, the entire algorithm fails.

We have utilized OpenCV for taking a running average of the current background for 30 frames and then using that average, we have detected the hand of the user. In order to recognize the hand in various different background settings, we have utilized background elimination.

Our network contains 7 hidden convolution layers with ReLu utilized as their activation function and 1 fully connected layer. The network is trained across 50 epochs with a batch size of 64. Increasing the number of epochs beyond 50 goes into the region of diminishing returns, where the overall training time increases greatly with no significant rise in accuracy. The model achieves accuracy of over 96% in the training dataset.

5 Results

We have successfully demonstrated the application of our project by coming up with two scenarios. In the first one, we control the volume of a song playing in the background using two gestures. In the second one, a much more complex scenario, we control the cursor on our computer using hand gestures. We have provided video demos for both the scenarios. Finally, we display the segmented hand region in the current frame and wait for a keypress to exit the program. Remember to update the background model by keeping the camera static without any motion. After 5-6 seconds, show your hand in the recognizing zone to reveal your hand region alone.

In the first scenario, we have mapped the volume up key to the gesture called "Swing". On the other hand, we mapped the volume down key to the gesture called "Swag". Both the gestures are visible in the provided demo.

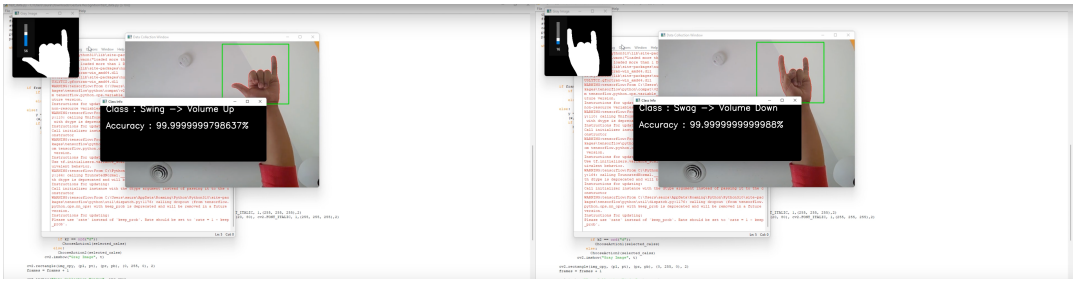

Figure 4: Controlling Volume via Gestures

Application 1 Demo Video

In our second application, we have mapped the movement of the cursor to gestures. We have mapped the move up command in the cursor to the gesture called "Palm", mapped the move right command in the cursor to the gesture called "Swag", mapped the move down command in the cursor to the gesture called "Swing", mapped the move left command in the cursor to the gesture called "Peace", mapped the click command in the cursor to the gesture called "Fist". All 5 gestures are visible in the provided demo.

Application 2 Demo Video 1

Application 2 Demo Video 2

Throughout the video, we see the cursor moving in accordance with the gestures explained above. Here, we have successfully presented some examples of gestures being utilized to carry out common tasks. The recent rapid development in the field of Deep Learning and Computer Vision have had a tremendous impact in making sure that such projects are possible. Our model achieves a high level of accuracy which is instrumental to carrying out the required task successfully. It is pretty obvious that the scope of this work is limitless.

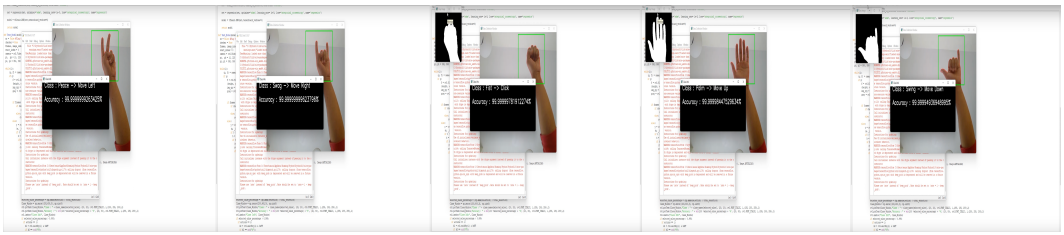

Figure 5: Moving Cursor via Gestures

6 References

1. P. Molchanov, S. Gupta, K. Kim and J. Kautz, "Hand gesture recognition with 3D convolutional neural networks," 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2015, pp. 1-7, doi: 10.1109/CVPRW.2015.7301342.

2. Hakim, N.L.; Shih, T.K.; Kasthuri Arachchi, S.P.; Aditya, W.; Chen, Y.-C.; Lin, C.-Y. Dynamic Hand Gesture Recognition Using 3DCNN and LSTM with FSM Context-Aware Model. Sensors 2019, 19, 5429. https://doi.org/10.3390/s19245429

3. M. Z. Islam, M. S. Hossain, R. ul Islam and K. Andersson, "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation," 2019 Joint 8th International Conference on Informatics, Electronics and Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision and Pattern Recognition (icIVPR), 2019, pp. 324-329, doi: 10.1109/ICIEV.2019.8858563.

4. Yu J, Qin M, Zhou S. Dynamic gesture recognition based on 2D convolutional neural network and feature fusion. Sci Rep. 2022 Mar 14;12(1):4345. doi: 10.1038/s41598-022-08133-z. PMID: 35288612; PMCID: PMC8921226.

5. Sahoo, J.P.; Prakash, A.J.; Pławiak, P.; Samantray, S. Real-Time Hand Gesture Recognition Using Fine-Tuned Convolutional Neural Network. Sensors 2022, 22, 706. https://doi.org/10.3390/s22030706