Function Based Hand Gesture Detection Using Image

(2021F) COMP-5112-FDE - Research Methodolody

Udhaya Kumar Rajendran  
*Department of Computer Science*  
Lakehead UniversityThunder Bay,Ontario

[rajendranu@lakeheadu.ca](mailto:rajendranu@lakeheadu.ca)

Vairag Nileshbhai Parikh   
*Department of Computer Science*  
Lakehead UniversityThunder Bay,Ontario

[vparikh2@lakeheadu.ca](mailto:vparikh2@lakeheadu.ca)

Guided By: Dr. Jinan Fiaidhi

*Department of Computer Science*  
*Lakehead University*Thunder Bay, Canada

[jfiaidhi@lakeheadu.ca](mailto:jfiaidhi@lakeheadu.ca)

*Abstract*— **In most modern problem-solving systems, image identification is becoming increasingly important. In glut, there are approaches for picture recognition, analysis, and classification, but the differences between them are still obscure. It is critical that appropriate distinctions between such strategies be evaluated and analyzed. The most important reason for the rise of gesture recognition is that it allows for an easy communication line between humans and computers, which is known as HCI (Human Computer Interaction). The purpose of this paper is to identify hand postures and develop a man- machine connection. From the learning phase, the hand region in the image is recognized, and the pattern is classified as the most comparable gesture. The input, which is an image or a frame from a video, can be collected from a web camera or any other camera in this method. This color image is transformed to a binary image and preprocessed, after which it is categorized using knowledge gained from prior models by a Neural Network. This is a straightforward and effective method.**

Index Terms— Gesture Recognition, Image Processing, Convolution Neural Network, Binary Image, Background Subtraction

# Introduction

As we all know, hand gesture detection using vision-based technologies is an important aspect of human-computer inter- action (HCI). In recent decades, the keyboard and mouse have become increasingly important in human-computer interaction. However, new sorts of HCI solutions have been necessary as a result of the rapid growth of technology and software. In the subject of HCI, technologies like speech recognition and gesture recognition garner a lot of attention.

As we all know, hand gesture detection using vision-based technologies is an important aspect of human-computer inter- action (HCI). In recent decades, the keyboard and mouse have become increasingly important in human-computer interaction. However, new sorts of HCI solutions have been necessary as a result of the rapid growth of technology and software. In the subject of HCI, technologies like speech recognition and gesture recognition garner a lot of attention.

A gesture is a visual representation of physical action or emotional expression. It consists of both body and hand gestures. There are two types of gestures: static gestures [1–2] and dynamic gestures [3–4]. A sign is denoted by the posture of the body or a hand gesture in the case of the

former. The latter sends messages through the movement of the body or the hand. Gesture can be used to communicate between humans and computers [5-6]. It differs significantly from traditional hardware-based techniques in that it allows for human-computer interaction via gesture recognition. The user intent is determined through gesture recognition, which recognizes the gesture or movement of the body or body components. Many academics have worked for decades to develop hand motion detection technology. Many applications, such as sign language recognition, augmented reality (virtual reality) [7–8], sign language interpreters for the disabled [9], and robot control [10, 11], rely heavily on hand gesture recognition.

We describe an efficient and successful approach for hand gesture recognition in this work. The background subtraction method is used to detect the hand region. The pattern of hand is then evaluated to classify in one of the similar category in knowledge base. Following the recognition of the fingers, the hand gesture can be categorized using a simple rule classifier.

# Literature Review

In the field of gesture and action detection, many hand- crafted spatio-temporal features for effective video analysis have been introduced [13, 14]. Image gradients and optical flow are commonly used to capture shape, appearance, and motion signals. For automobile gesture recognition, Ohn-Bar and Trivedi [15] assess a number of global features. Improved dense trajectories [16] and Fisher vector [17] representations, which are commonly regarded as state-of-the-art local features and aggregation techniques for video analysis, are successfully used in a variety of video classification systems. Depth sensors frequently have features that are tailored to the individual properties of the depth data. Random occupancy patterns [18], for example, make use of point clouds, while super normal vectors [19] make use of surface normals.

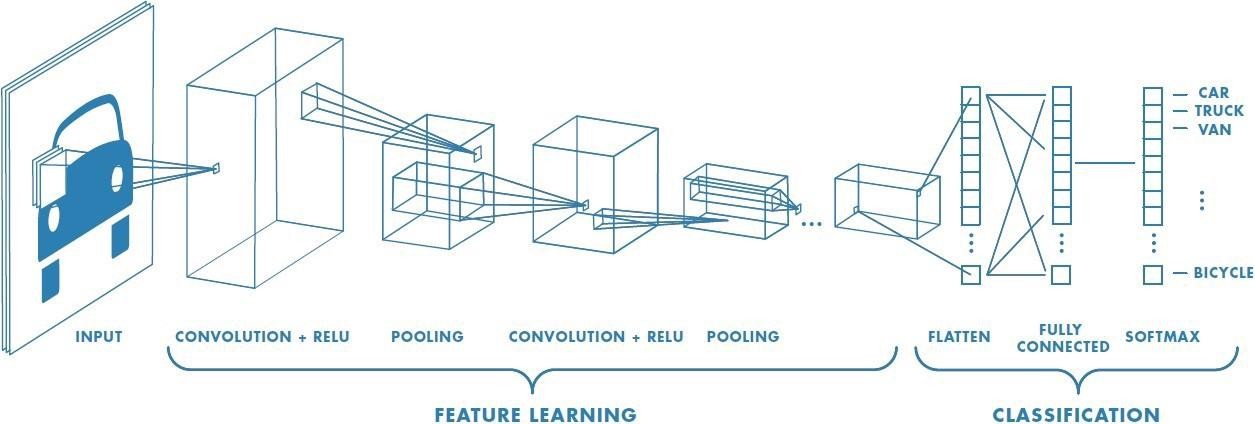


Fig. 1. Working of CNN

In contrast to hand-crafted features, deep neural networks are increasingly being used to develop feature representations. Neverova et al. [20] use CNNs to recognize sign language gestures by combining color and depth data from hand regions and upper-body skeletons. To minimize overfitting, Molchanov et al. [21] apply a 3D-CNN to the entire video sequence and incorporate space-time video augmentation approaches. Simonyan and Zisserman [22] suggest separate CNNs for the spatial and temporal streams that are late-fused and explicitly exploit optical flow in the context of action recognition. Tran et al. [23] use a 3D-CNN to assess a set of short video clips and average the responses of the network over all clips. The majority of earlier approaches either used pre-segmented video sequences or approached detection and classification as separate issues.

# Methodology

CNN is a neural network-based architecture that can recognize and classify specific features in images, and it is commonly employed for image analysis. Image and video recognition, image classification, medical image analysis, computer vision, and natural language processing are only a few of their uses. CNN in image recognition deals with pixel data i.e., the image is represented in the form of matrix. The CNN architecture primarily consists of convolution layers that helps in the process of feature extraction. The images are represented in the form of matrices. These matrices are multiplied with kernel filters of a pre-determined size to extract features from the image.

In CNN architecture, there are three different layers: The convolutional layers, pooling layers and fully-connected layers. Convolutional layer is the initial layer that extracts the different features from the input photos. The convolution mathematical operation is done between the input image and a filter of a specific size MxM in this layer. The dot product between the filter and the sections of the input image with regard to the size of the filter is taken by sliding the filter across the input image (MxM). The Feature map is the result, and it contains information about the image such as its corners and edges. This feature map is then supplied to further layers, which learn a variety of other features from the input image.

A Pooling Layer is usually applied after a Convolutional Layer. This layer's major goal is to lower the size of the convolved feature map in order to reduce computational expenses. This is accomplished by reducing the connections between layers and operating independently on each feature map. There are numerous sorts of Pooling operations, depending on the mechanism utilized. The largest element is obtained from the feature map in Max Pooling. The average of the elements in a predefined sized Image segment is calculated using Average Pooling. Sum Pooling calculates the total sum of the elements in the predefined section. The Pooling Layer is typically used to connect the Convolutional Layer and the FC Layer.

The weights and biases, as well as the neurons, make up the Fully Connected (FC) layer, which is used to connect the neurons between two layers. The last several layers of a CNN Architecture are usually positioned before the output layer. The previous layers' input images are flattened and supplied to the FC layer in this step. After that, the flattened vector is sent via a few additional FC layers, where the mathematical functional operations are normally performed. The classification procedure gets started at this point.

When all of the features are connected to the FC layer, the training dataset is prone to overfitting. Overfitting happens when a model performs so well on training data that it has a negative impact on its performance when applied to new data. To address this issue, a dropout layer is employed, in which a few neurons are removed from the neural network during the training process, resulting in a smaller model. After passing a dropout of 0.3, 30% of the nodes in the neural network are dropped out at random.

Finally, the activation function is one of the most crucial elements in the CNN model. They're utilized to learn and approximate any form of network variable-to-variable association that's both continuous and complex. In simple terms, it determines which model information should fire in the forward direction and which should not at the network's end. It gives the network non-linearity. The ReLU, Softmax, tanH, and Sigmoid functions are some of the most often utilised activation functions. Each of these functions has a distinct use. For a binary classification CNN model, sigmoid and softmax functions are favored, whereas softmax is typically employed for multi-class classification.

*A. Dataset*

We have used 6 gestures for training and testing the built convolutional neural network model. There are 1200+ images in each of the gesture category for training the model.

Following are the 6 gestures.

|  |  |
| --- | --- |
| **Image** | **Gesture** |
|  | Fist |
|  | L |
|  | OK |
|  | Palm |
|  | Thumbs Up |
|  | Thumbs Down |

Table 1: Gesture dataset description

The input dataset is split into training data and testing data respectively. It is essential to keep the data manageable especially when working with large dataset for a machine learning project. Pytorch supports preparing custom dataloader with the dataset we have which can be batched and iterated while training the deep network model. The images in the resultant dataset are converted to their corresponding pixels. The label is extracted by fetching the name of the folder, the image is stored in for each category. Through repetitive iterations, every image in the respective folders is converted to pixels and associated with the name of the folder as labels.

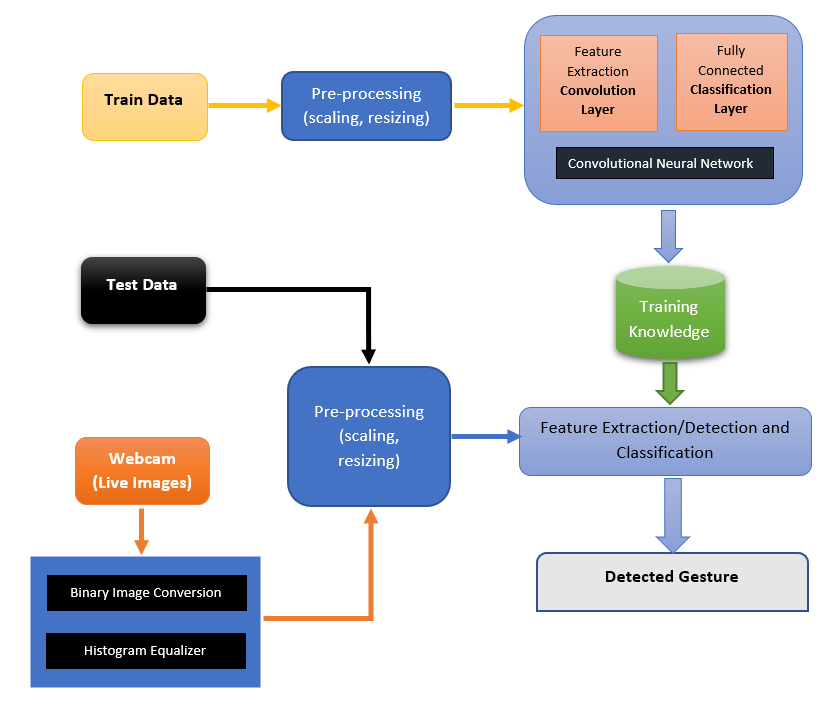


Fig 2: System Diagram

The images are loaded using the torchvision package. Here, the images are transformed such as the images can be scaled and resized. Now, the pixel of an image is considered as X and the folder name is considered as y.

Below are the parameters used for the building Convolutional Neural Network in this project.

|  |  |
| --- | --- |
| **Parameter** | **Value** |
| Input Channels | 1 (binary image) |
| Kernel Size | 3 |
| Stride | 1 |
| Padding | 1 |
| Dropout | 0.2 |
| Pooling | MaxPooling(Kernel size=2) |
| Batch Size (Training) | 32 |
| Learning Rate | 0.01 |
| Number of epochs (iterations) | 3, 10, 25, 50 (different combinations) |
| Loss Function | Cross Entropy Loss |
| Optimizer | Adam |

Table 2: Hyper parameters for CNN

For every epoch,

a. The training data is allowed to pass through the convolution layers which extract the features and the flatten layer will convert the m\*n\*d array to one dimension vector which is passed to the fully connected dense layer. Finally, the output layer will produce the output values which is correlated with the class label of an image.

b. The predicted labels are compared with the original target labels and the loss is calculated with the help of Loss function.

c. The loss is back propagated in the network so that the weights and bias in the layers of the deep network gets adjusted to minimize the loss function.

The trained model can be saved and the same trained model can be loaded next time which will use the available learned weights (from the training) for prediction.

In case of processing live gestures, With the help of OpenCV-Python library, every frame captured from webcam is converted, pre-processed, and then allowed to utilize the knowledge of the trained model for prediction. The frame that is captured via webcam is converted to binary image followed by histogram equalization. The image is then resized and converted to pixels before using it in convolution layers. The testing phase is same as mentioned before.

# Results

The exploratory outcomes have been appeared and the investigation of results has been done in this segment. The execution of the classifiers is assessed dependent on three parameters to be Classification Accuracy (CA), precision and recall.

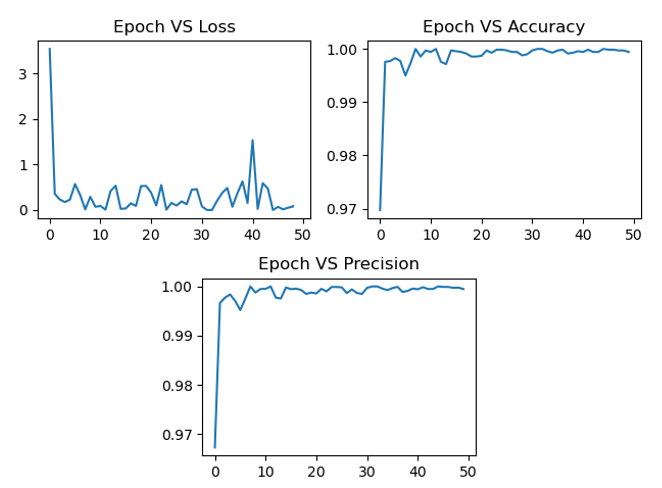
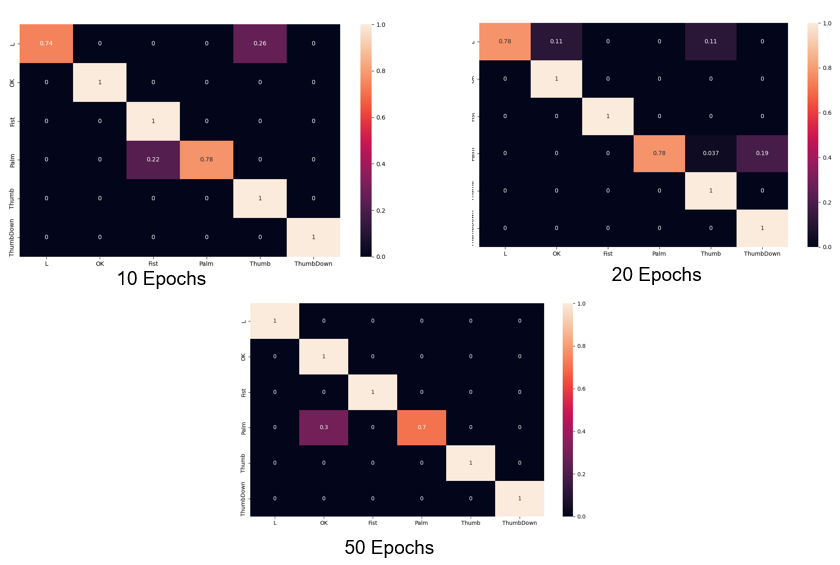
The model was trained for 50 epochs in total. There should be a minimum amount of iteration that the training samples should go through the model in order for the model to adjust itself for better performance. 

Fig 3: Training Loss, Accuracy and Precision for 50 epochs

Fig 4: Confusion matrix for 10 epochs, 20 epochs and 50 epochs

The training is carried out for 50 epochs and the training accuracy, precision and loss is fetched for every epoch. For the first few epochs, the accuracy is dynamic but after few iterations, the accuracy and precision are stabilized. Similarly, the training loss is reduced eventually over few epochs. The confusion matrix is obtained for the model over 10, 20 and 50 epochs. For example, the confusion matrix obtained after the first 10 epochs, it is noted that the gesture L is identified 0.74 times. Whereas, when the model is trained with 50 epochs, the model prediction increases.

# Conclusion and future work

This paper proposes an automatic hand gesture spotting for various gesture used in American Sign Language and General Gesture used in daily life using image processing. Our experiment performs the hand gesture spotting and recognition tasks using neural network that can be further expanded by increasing the knowledge base of model. Furthermore, it is suitable for real-time applications and solves the issues of background noise through background cancellation from live stream video. The results show that the proposed method can successfully recognize isolated gestures with 96.51% and spotting meaningful gestures that are embedded in the input video stream with 90.49% reliability. The future research will improve hand gesture spotting accuracy using short gesture detector and fingertip detection for gesture path in conjunction with multicamera system.

# References

|  |  |
| --- | --- |
| [1] | A. D. Bagdanov, A. Del Bimbo, L. Seidenari, and L. Usai, “Real-time hand status recognition from RGB-D imagery,” in Proceedings of the 21st International Conference on Pattern Recognition (ICPR ’12), pp. 2456–2459, November 2012. |
| [2] | M. Elmezain, A. Al-Hamadi, and B. Michaelis, “A robust method for hand gesture segmentation and recognition using forward spotting scheme in conditional random fields,” in Proceedings of the 20th Inter- national Conference on Pattern Recognition (ICPR ’10), pp. 3850–3853, August 2010. |
| [3] | P. Suryanarayan, A. Subramanian, and D. Mandalapu, “Dynamic hand pose recognition using depth data,” in Proceedings of the 20th Interna- tional Conference on Pattern Recognition (ICPR ’10), pp. 3105–3108, August 2010. |
| [4] | S. Park, S. Yu, J. Kim, S. Kim, and S. Lee, “3D hand tracking using Kalman filter in depth space,” Eurasip Journal on Advances in Signal Processing, vol. 2012, no. 1, article 36, 2012.. |
| [5] | M. Panwar, “Hand gesture recognition based on shape parameters,” in Proceedings of the International Conference on Computing, Com- munication and Applications (ICCCA ’12), pp. 1–6, February 2012. |
| [6] | 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. |
| [7] | D. Wickeroth, P. Ben o¨lken, and U. Lang, “Markerless gesture based interaction for design review scenarios,” in Proceedings of the 2nd International Conference on the Applications of Digital Information and Web Technologies (ICADIWT ’09), pp. 682–687, August 2009. |
| [8] | V. Frati and D. Prattichizzo, “Using Kinect for hand tracking and rendering in wearable haptics,” in Proceedings of the IEEE World Haptics Conference (WHC ’11), pp. 317–321, June 2011. |
| [9] | J. Zeng, Y. Sun, and F. Wang, “A natural hand gesture system for intelligent human-computer interaction and medical assistance,” in Pro- ceedings of the 3rd Global Congress on Intelligent Systems (GCIS ’12), pp. 382–385, November 2012. |
| [10] | D. Droeschel, J. St u¨ckler, and S. Behnke, “Learning to interpret point- ing gestures with a time-of-flight camera,” in Proceedings of the 6th ACM/IEEE International Conference on Human-Robot Interaction (HRI ’11), pp. 481–488, March 2011. |
| [11] | K. Hu, S. Canavan, and L. Yin, “Hand pointing estimation for human computer interaction based on two orthogonal-views,” in Proceedings of the 20th International Conference on Pattern Recognition (ICPR ’10), pp. 3760–3763, August 2010. |
| [12] | Zhi-hua Chen, Jung-Tae Kim, Jianning Liang, Jing Zhang and Yu-Bo Yuan, “Real-Time Hand Gesture Recognition Using Finger Segmenta- tion” in Volume 2014 —Article ID 267872 on 25 June 2014. |

|  |  |
| --- | --- |
| [13] | X. Shen, G. Hua, L. Williams, and Y. Wu. Dynamic hand gesture recognition: an examplar based approach from motion divergence fields. Image and Vis. Computing, 2012. |
| [14] | A. Tamrakar, S. Ali, Q. Yu, J. Liu, O. Javed, A. Divakaran, H. Cheng, and H. Sawhney. Evaluation of low-level features and their combinations for complex event detection in open source videos. In CVPR, 2012. |
| [15] | E. Ohn-Bar and M. Trivedi. Hand gesture recognition in real time for automotive interfaces: a multimodal vision-based approach and evaluations. IEEE ITS, 15(6):1–10, 2014. |
| [16] | H. Wang, D. Oneata, J. Verbeek, and C. Schmid. A ro- bust and efficient video representation for action recognition. IJCV, 2015. |
| [17] | F. Perronnin, J. Sanchez, and T. Mensink. Improving the fisher kernel for large-scale image classification. In ECCV, 2010. |
| [18] | J. Wang, Z. Liu, J. Chorowski, Z. Chen, and Y. Wu. Robust 3D action recognition with random occupancy patterns. In ECCV, 2012. |
| [19] | X. Yang and Y. Tian. Super normal vector for activity recog- nition using depth sequences. In CVPR, 2014. |
| [20] | N. Neverova, C. Wolf, G. W. Taylor, and F. Nebout. Multi- scale deep learning for gesture detection and localization. In ECCVW, 2014. |
| [21] | P. Molchanov, S. Gupta, K. Kim, and J. Kautz. Hand ges- ture recogni- tion with 3D convolutional neural networks. In CVPRW, 2015. |
| [22] | K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition. In NIPS, 2014. |
| [23] | D. Tran, L. D. Bourdev, R. Fergus, L. Torresani, and M. Paluri. Learning spatiotemporal features with 3D con- volutional networks. In ICCV, 2015. |