Gesture Recognition using Convolutional Neural Network

Pavithran Palanichamy
Electrical and Computer Engineering
University of British Columbia

***Abstract*—With the development of today's technology, and as humans tend to naturally use hand gestures in their communication process to clarify their intentions, hand gesture recognition is considered to be an important part of Human-Computer Interaction (HCI). Hand gesture recognition systems are widely distributed on different kinds of applications, ranging from applications such as sign language detection, robotics, augmented reality and medical systems. This project presents a system to detect the hand gestures form a live video feed using the average background algorithm for background subtraction and convolutional neural network for classification. The designed convolutional model is trained with three different datasets and the performance of each model is compared using the real-time video feed. The designed gesture recognition system has an accuracy of about 99% for larger datasets.**

***Index Terms*—Hand Gesture Recognition, Human-Computer Interaction, Computer Vision, Background subtraction, Convolutional Neural Networks**

I. Introduction

With the development of information technology, computer systems have become an integral part of human life. This has led researchers and engineers to rethink the ways to interact with computers. Hand gestures provide an attractive alternative to the traditional user interface devices such as keyboard, mouse and joystick for Human-Computer Interaction (HCI). Hand gestures are simple, natural and are universally understood.[2]. In Human-Computer Interaction (HCI), hand gestures guarantee the speed of communicating with the computer, give a user-friendly and aesthetic environment, provide a contactless interface with the computer for comfort and safety of users, and makes the control of complex and virtual environments much easier [10]. Thus, gesture recognition has been used in a wide range of applications such as communication tool between hearing impaired, gesture-controlled robots, augmented reality applications and even medical applications. Researches in [14], [15] have established the importance of gesture recognition systems for Human Computer Interaction.

Hand gesture recognition has been active research area for more than 40 years [16]. In 1977, Zimmerman proposed a system that detects the number of fingers bent in a hand. Since then, the field of hand gesture recognition has been growing and a large amount of work is being conducted in the last 2 to 3 years due to developments in image acquisition hardware, computer vision and machine learning algorithms.

Two approaches are commonly used in hand gesture recognition systems. The First approach employs sensors (mechanical or optical) attached to a glove that transduces finger flexions into electrical signals for determining the hand posture [3]. The other approach uses computer vision-based techniques that are contactless and based on the way human beings perceive information about their surroundings. Vision-based hand gesture recognition to extract images were studied in was proposed by Weiguo et al. [17] and Ananyaa et al. [18]. A good vision-based hand gesture recognition system must be robust, computationally efficient, and scalable. Many hand gesture recognition systems proposed previously detect and segment the hand gestures from the background using motion detection or skin colour. Proper selection of features and sophisticated recognition algorithms, can affect the success or failure of the hand gesture recognition system [3] .Results from these researches have shown that the vision-based technique was more stable and reliable compared to the data glove-based technique.

Generally, hand gesture recognition systems can be divided into 4 stages as shown in 1: (i) Image acquisition, (ii) Hand tracking, (iii) Classification, and finally (iv) Output gesture [5]. The first step of image acquisition or gesture acquisition is to capture the hand gestures by the computer. This can be achieved by using image acquisition devices such as web camera, gloves that detect the movements of the user, and motion sensing input devices captures the hand gestures and motions. The next step of hand tracking is done in order to trace the user's hand and separate it from the background and from the surrounding objects. Several algorithms such as single Gaussian, Kalman filter, Mixture of Gaussians (MoG), clustering-based and Hidden Markov Models (HMM) can be employed for this purpose. The image extracted from the previous steps are used to train the classifiers. Various classifiers such as Artificial Neural Networks (ANN), K-nearest neighbour (KNN), Naive Bayes (NB), Support Vector Machine (SVM), etc. can be used for predicting the output gesture.

In this project, a vision-based hand gesture detection system using the average background algorithm for background subtraction and convolutional neural network for classification is designed. The model is tested and compared with three different datasets including a custom generated dataset with 7 classes. The first contribution is the use of average background algorithm to the live video stream to predict the gestures from the dataset over 95% accuracy. The second contribution is a comparative analysis of the performance of the designed model

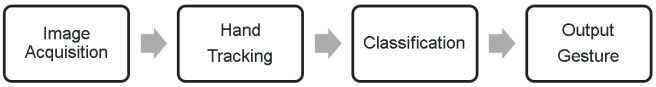

Fig. 1. Basic steps of hand gesture recognition system.

across different datasets. The rest of the paper is organized as follows. Section II discusses the methods used in this project which includes background subtraction and contour detection. The details of the dataset used are discussed in Section III. Section IV explains the architecture of the convolutional neural network. Section V provides the results and performance of the system followed by the conclusion in Section VI.

II. Methods

Figure. 2 shows the block diagram of the gesture recognition system used in this project. This system consists of two main stages. The first stage of the system is the pre-processing stage which consists of background subtraction and contour extraction. This process is carried out to remove the static objects in the background and to extract the region of hand from the live video stream. The second stage is the gesture detection stage. The pre-processed image from the previous stage is fed as an input to a convolutional neural network (CNN). The CNN is trained initially with a labelled dataset until convergence. The trained weights are stored at the end of the training. Finally, the trained model is used for predicting hand gestures from the real-time video frame input.

A. Background Subtraction Algorithm

Background subtraction methods are well established and widely used to detect moving objects from static cameras. The main principle of this approach is to detect moving objects by subtracting the current frame from a reference frame, which is called a "background image". This background image is a representation of the scene without moving objects acquired during the initialization phase [11]. Disturbances such as light changes, shadows, overlapping of the objects in the visual area are taken into account by continuously updating the background model. There are various approaches for background subtraction such as frame difference, Gaussian mixture model, kernel density estimation and codebook. The choice of the method depends on the performance requirements of the application. In this project, running gaussian average [1], [12] is chosen for background subtraction because it is simple, accurate and has low memory requirements. This approach is divided into three main parts: (i) background modelling, (ii) foreground detection(background subtraction) (iii) updating the background model. An initial background from the first N frames ($\mathrm{F}(i)_{(x,y)}$) is modelled using an average filter ($\mathrm{A}_{(x,y)}$) as shown in (1). Secondly, the foreground objects are obtained by subtracting the current frame from the average background as given in (2). After this, a threshold filter is applied to create a binary image ($\mathrm{B}_{(x,y)}$) as calculated in (3). The threshold value is chosen in such a way that the background and the foreground are distinguishable. Finally, the average background's pixels are updated using (4) to remove any noise or any static objects. The parameter α regulates the update speed; in other words, it controls how fast the average background is updated by the current frame.

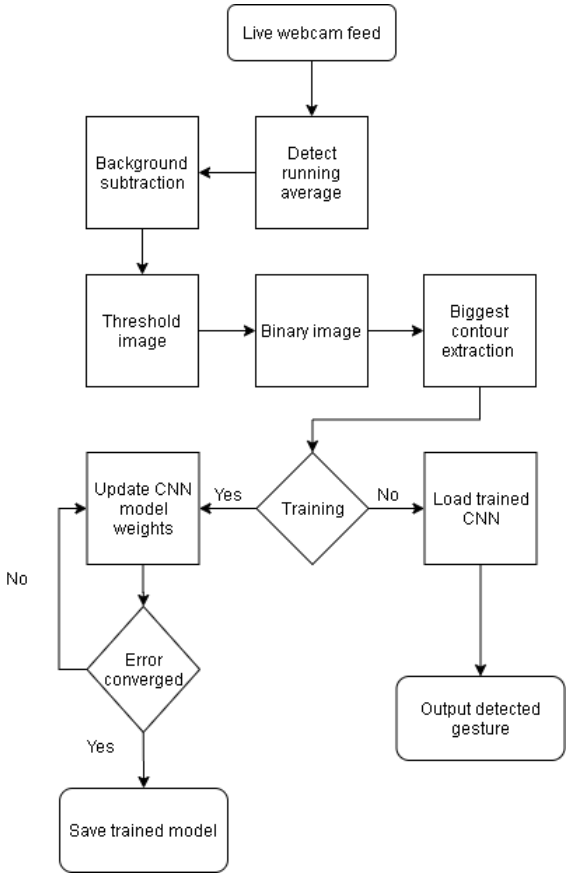

Fig. 2. System block diagram.

$$A_{(x,y)} = \frac{1}{N}\sum_{i=1}^{N} F(i)_{(x,y)} \tag{1}$$

$$\Delta \mathrm{A}_{(x,y)} = |F_{(x,y)} - \mathrm{A}_{(x,y)}| \tag{2}$$

$$B_{(x,y)} = \begin{cases} 1, & \Delta A_{(x,y)} > Threshold \\ 0, & Otherwise \end{cases} \tag{3}$$

$$A_{(x,y)} = \alpha \cdot F_{(x,y)} + (1-\alpha) \cdot A_{(x,y)} \tag{4}$$

B. Contour Extraction Algorithm

A curve that connects the regions of an image that has the same colour or intensity is called a contour. The foreground binary image obtained from the background subtraction might still contain noises in the background. This could be caused due to objects in the background having colour intensity similar to the foreground object, change in brightness and overlap by other objects. In order to separate only the region

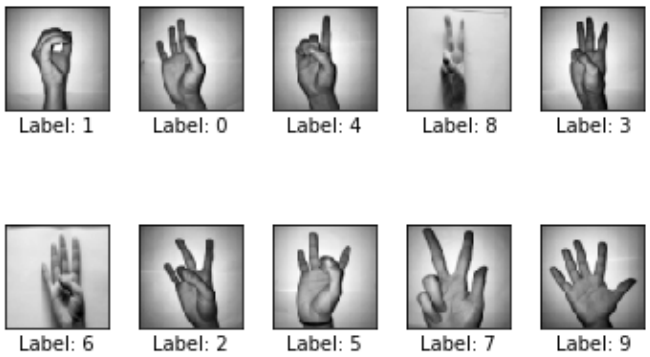

Fig. 3. Sign language digits dataset

of interest Suzuki's algorithm [13] is used to find the biggest contour of the thresholded image. The biggest contour, which has the largest calculated area represents the hand contour.

III. Dataset

The convolutional neural network must be trained with labelled images in order to predict the gesture. It is important to select an appropriate dataset for training because the performance of CNN directly depends on it. The general requirements for any dataset are that the individual data are reasonably distinct with variations. A large number of samples is required to generalize the model and to avoid over-fitting. In this project 3 different datasets are used for training and performance of the model is compared with a test dataset.

A. Sign language digits dataset

The first dataset used is the sign language digits dataset created by students of Turkey Ankara Ayrancı Anadolu High School [19]. The dataset is made available online in Kaggle's website. The dataset contains 10 different classes with 2062 images in total. Each class represents digits from 0 to 9 in sign language as shown in Figure. 3. This dataset has a considerable amount of variations as it is collected from 10 different subjects with 10 samples each. The data is stored as 64 x 64 grayscale image in .npy file format. The dataset is split into training, validation and test sets each containing 1649, 206 and 207 images respectively.

B. Hand gesture recognition database

The second dataset is the hand gesture recognition database [20], composed by a set of near-infrared images acquired by the Leap Motion sensor. This is a huge dataset with almost 2 GB of data and is available online in Kaggle's website. The database is composed of 10 different hand-gestures as shown in 4 that were performed by 10 different subjects (5 men and 5 women). It consists of 20,000 labelled hand gestures. The reason for choosing this dataset is because it is one of the largest and widely used dataset for gesture recognition models. The data is stored as 224 x 224 grayscale images in .png format. Out of the 20000 images, 16000 are used as the training dataset, 2000 each for validation and test dataset.

TABLE I
Dataset partition

Dataset	Training Data	Validation Data	Testing Data
Sign language digits	1649	206	207
Hand gesture recognition	16000	2000	2000
Custom	21000	4200	100

C. Custom dataset

There is a disconnect between the previous two datasets and the image obtained from the live webcam stream. To achieve better performance, the training data and the testing data both must be from the same image acquisition method. Hence, a custom dataset was created to train the CNN. OpenCV, an open-source computer vision library, was used to create the dataset. A custom dataset is created as shown in Figure. 5.The database is composed of 7 different hand-gestures that were performed by 4 different subjects (3 men, 1 woman). Background, orientation and brightness were changed in order to have variations between the data. It consists of 21,000 labelled hand gestures. The data obtained from the webcam feed is extracted using the techniques explained in Section. II and is stored as 89 x 100 grayscale images in .png format. 21000 images are used for the training set, 4200 for validation set and 100 for test data.

IV. Convolutional Neural Network

The convolutional neural network (CNN) is a class of deep learning neural networks. A CNN works by extracting features from images which eliminate the need for manual feature extraction. The features are learned while the network trains on a set of images which makes deep learning models extremely accurate for computer vision tasks. Each layer increases the complexity of the learned features. A CNN convolves learned features with input data and uses 2D convolutional layers making it ideal for processing 2D images. Hence, CNN has been chosen as the classifier for detecting the hand gestures.

The architecture of CNN used for the purpose of classification is depicted in the Figure. 6. It contains 6 convolutional layers and 3 fully connected layers. Each convolutional layer is followed by a MaxPool layer. All the layers use ReLU activation function. Softmax function at the output layer produces the probability of each gesture class at each node. The width of each convolutional layer in the figure 6 represents the number of filters used in that layer, while the numbers on the red and blue squares show the size of each filter.

Regularization is used to prevent over-fitting. The dense layers of the CNN use dropout and early stopping is used to prevent over-fitting and to save time. Training is conducted using a categorical cross-entropy loss function and using the famous Adam optimiser. The configurations of the CNN is summarized in II. The network was implemented and trained in Keras running on top of TensorFlow on an Intel Core i5-8250U CPU @1.60 GHz and 8 GB RAM without any GPUs. The maximum runtime for training is 58 minutes.

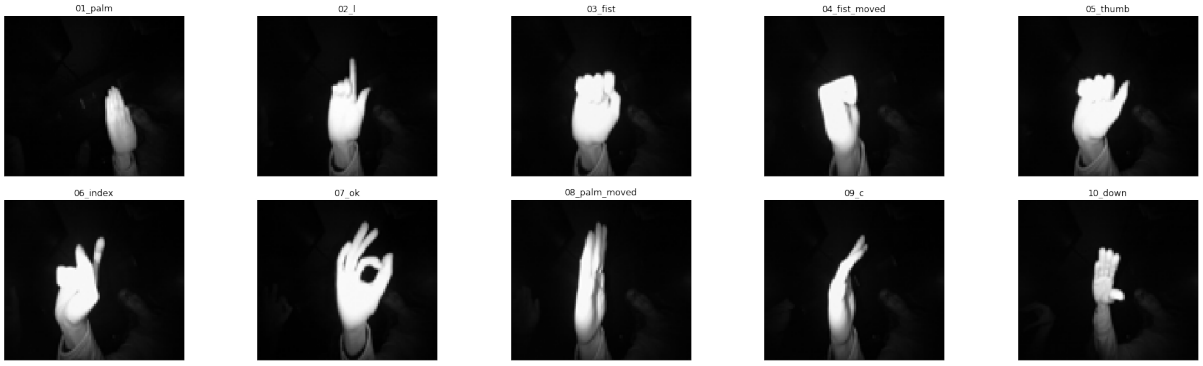

Fig. 4. Hand gesture recognition database

Fig. 5. Custom dataset

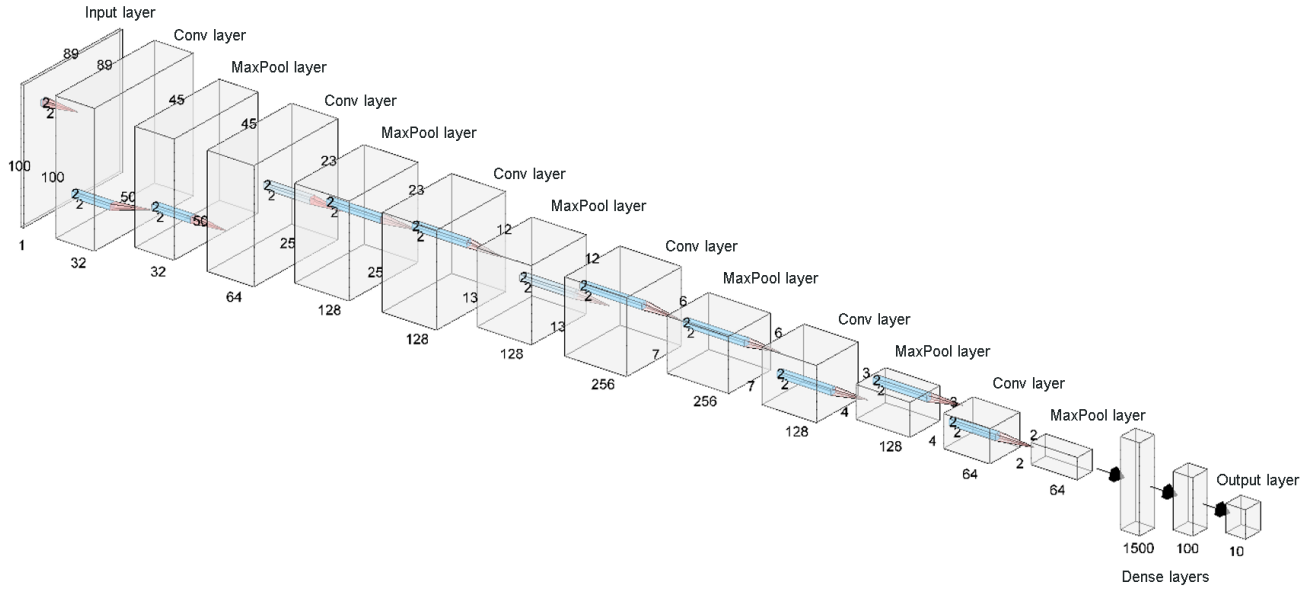

Fig. 6. CNN architecture

TABLE II
CONFIGURATIONS OF THE CNN

Parameters	Value
Number of convolutional layers	6
Number of Maxpool layers	6
Dropout rate	0.5
Activation function of hidden layer	ReLU
Activation function of output layer	SoftMax
Maximum number of epochs	100
Cost function	Categorical cross-entropy
Training optimiser	ADAM

V. EXPERIMENTAL SETUP

Several software packages were used in this project. Python3 has been used for code development and Keras with Tensorflow backend was used to create and train the CNN. Opencv is used for background elimination and contour detection algorithm. The other software libraries used in this project are Numpy, Pillow (PIL), Imutils and Sklearn. The training code is saved as a Jupyter notebook (.ipynb) and preprocessing and gesture detection codes are saved as python file (.py).

All three datasets are trained individually and tested with the same CNN architecture as shown in Figure. 6. Often times in machine learning if a large and more diverse dataset is present, a better model can be built. One way to getting a larger dataset is by creating new data by image augmentation methods. In this project augmentation techniques such as random rotation, width shift, height shift and zoom were used. The training is carried out until the error rate converges to a minimum value. The training progress is monitored by tracking the training and validation loss at the end of each epoch. The performance of the model with test data is interpreted in the form of a confusion matrix. The best weights are then stored at the end of the training.

The real performance of the system depends on how it works with the live video stream. The live webcam feed is obtained using OpenCV, an open-source computer vision library. Once the frame has been captured, the image in the region of interest is separated. The RGB image is converted to a grayscale image and the background is modelled using the running average method described in Section II. The foreground object is obtained by background subtraction and contour extraction. The image is resized to match CNN's input layer dimensions. With this extracted image, CNN predicts the hand gesture using the previously trained weights. The predicted gesture and confidence levels are displayed on a separate window continuously.

The background subtraction algorithm, contour extraction and running averages are available as functions in the OpenCV software library which makes the preprocessing of the video frame simpler. Tensorflow's Keras library is a powerful tool for designing and training CNN. The inbuilt optimization and image augmentation algorithms are used to create a complex model in concise and clear codes.

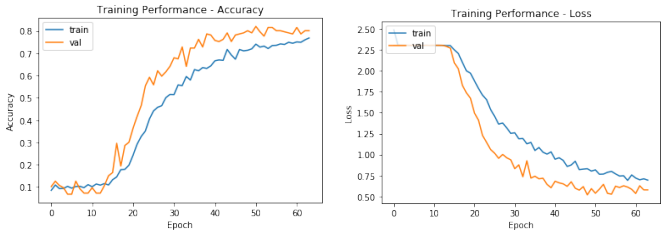

Fig. 7. Training performance of Sign language digits dataset

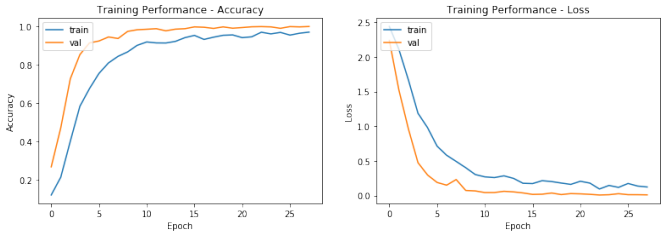

Fig. 8. Training performance of Hand gesture recognition database

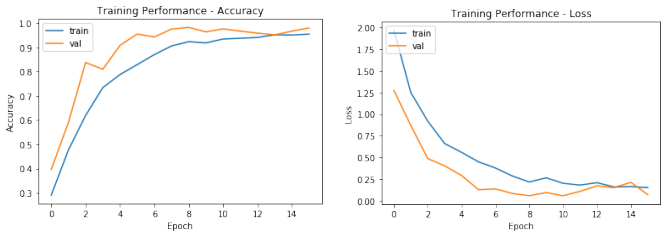

Fig. 9. Training performance of custom dataset

VI. RESULTS

The CNN was able to predict the gestures in all the datasets accurately. This is because of the many convolutional layers which extract features in the image. The metric used to measure the performance of the network are categorical cross-entropy loss and accuracy. Figure. 7, 8 and 9 shows the training performance of the three datasets. The validation accuracy for the sign language digits reached only 80.10%, whereas for Hand gesture recognition and the custom dataset it reached 99.80% and 98.00%. This shows the ability of CNN to adapt to different training datasets and perform consistently with high accuracy. The plots contain the accuracy and cross-entropy loss of both training and validation data. The validation loss is always less than the training loss which indicates the model is not overfitted. Augmentation of the images and early stopping has helped to avoid overfitting.

Table. III summaries the performance metrics of CNN for the three datasets. From the table, it can be noticed that the sign language digit dataset has the least performance. There are several reasons for this: (i) Small dataset and (iii) Improper lighting and (iii) similarity between images of different classes. The dataset of sign language digit dataset is very small compared to the other two datasets which lead to overfitting and restricts the CNNs ability to generalize. Improper lighting

TABLE III
PERFORMANCE METRICS OF THE CNN FOR DIFFERENT DATASETS

	Metric	Sign language digits	Handgesture recognition	Custom
Training data	Loss	0.69	0.12	0.15
	Accuracy	0.77	0.97	0.95
Validation data	Loss	0.58	0.01	0.07
	Accuracy	0.80	0.99	0.98
Test data	Loss	0.63	0.01	0.02
	Accuracy	0.79	0.99	0.99

creates a problem during the preprocessing stage. parts the skin and background are in similar colour intensities, the contour extraction algorithm assumes the background as part of the hand. This distorts the image segment and leads to improper classification. Finally, the gesture for digit three, zero, two, five and seven all have 3 fingers elongated and two fingers folded. This similarity between different classes affects the performance of the network to some extent.

The accuracy CNN with each dataset can be interpreted in the form of a confusion matrix. It allows the visualization of the performance of CNN. Figure. 10 shows the performance of the CNN with sign language digits dataset. The plot has a strong diagonal which signifies the accuracy of the model and it can be noticed that most of the labels were correctly classified. It is interesting to note that almost all classes except digit zero have some images misclassified as digit three. The reason for this is explained in point (iii) in the previous paragraph.

Figure. 11 shows the performance of the CNN with Hand gesture recognition database. The confusion matrix has a strong diagonal and all the classes are classified correctly almost every time. Similarly, Figure. 12 shows the performance of the CNN with a custom dataset which has a strong diagonal in the confusion matrix. The high accuracy for these two models is because of the large dataset, uniformly distributed images across classes and reasonable differences between the gestures. One important reason is that the background doesn't have any objects with the same intensity of the hand region which otherwise would affect detection of hand contours.

After the finalising the weights for the convolutional neural network, the testing was carried out with the live video feed. Though the running average algorithm for modelling the background is simple, it works very well. The preprocessed image frame from the webcam is used as the input to the CNN, which in turn predicts the gesture. Figure. 13, 14 and 15 shows the prediction of the live camera feed with the CNN trained with different datasets. The leftmost image in all these figures shows the image from captured by the webcam and the detected contour is overlaid on top of the hand segment. The middle images show the extracted foreground binary image after the preprocessing stage which is used as the input to the CNN. The leftmost image shows the predicted gesture along with the confidence level. These results show that the foreground binary image is correctly segmented from the live video feed and the CNN predicts the gestures with high accuracy.

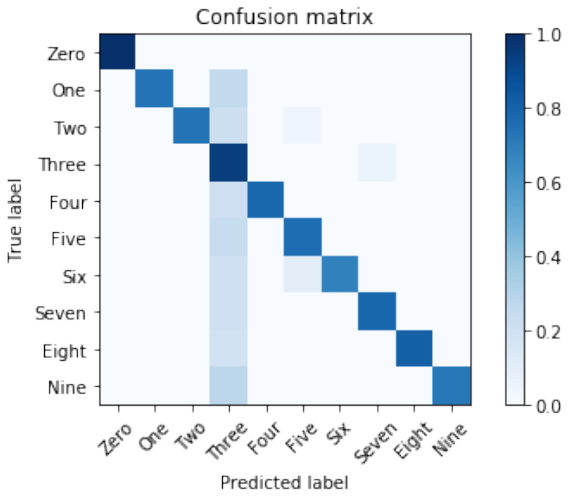

Fig. 10. Accuracy for Accuracy for Sign language digits dataset

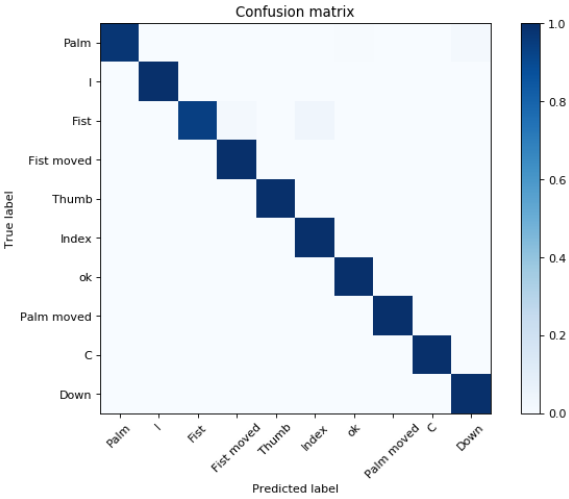

Fig. 11. Accuracy for Hand gesture recognition database

A. Limitations

The designed model is not without limitations. The algorithms used for preprocessing did not perform well with some images in the sign language digit dataset. Figure. 16 shows the problems of with the contour extraction and binary thresholding. Since few images contain regions in the background that have a similar intensity to that that of the hand, some parts of the background is also getting segmented as the hand region. This results in distorted noisy image and results in poor performance during classification. Secondly, since the resolution of the images from the dataset is low, the detected contours are not smooth. These problems can be solved by choosing an alternate contour extraction algorithm which does not take background intensities into account while computing the hand segment.

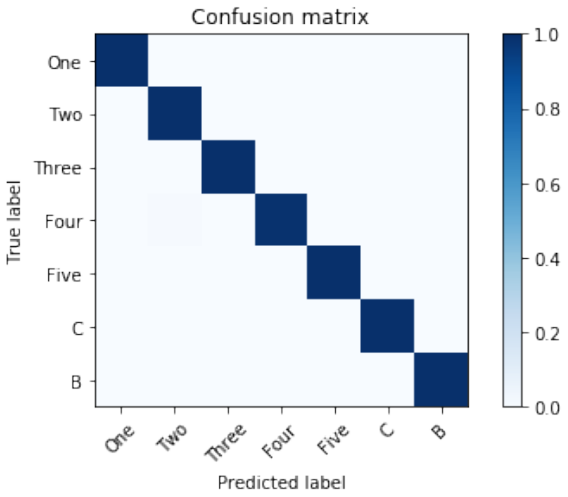

Fig. 12. Accuracy for custom dataset

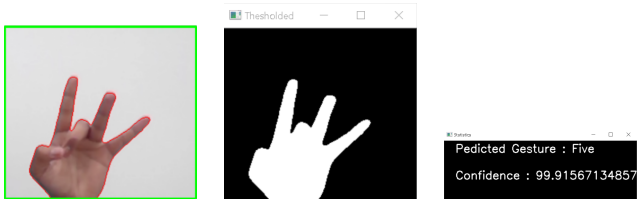

Fig. 13. Prediction of Sign language digits dataset

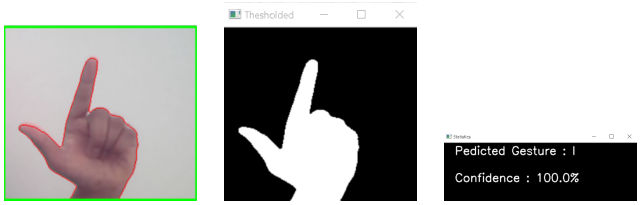

Fig. 14. Prediction of Hand gesture recognition database

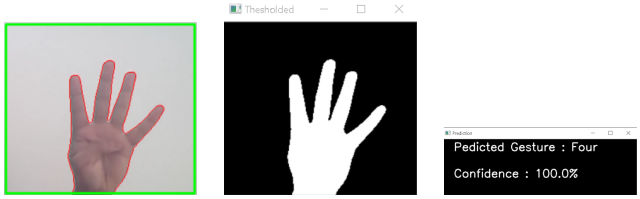

Fig. 15. Prediction of gestures in custom dataset

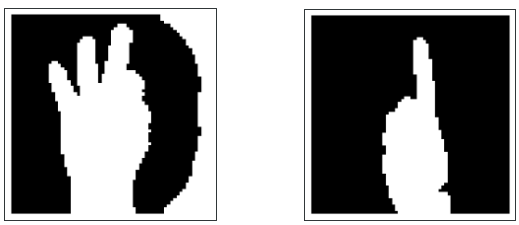

Fig. 16. Limitations with sign language digit dataset

VII. Conclusion

This project presented a system for hand gesture detection by using the average background algorithm for background subtraction and convolutional neural network as the classifier. The system is designed to detect hand gestures from a real-time video stream. The model was experimented with three different datasets and results show that it performs consistently well all the datasets. The results show that the designed model can achieve 99.0% accuracy with larger datasets. Experiments conducted with the trained model showed that the detection has very low latency and the performance is invariant to changes in brightness, orientation and background. However, the contour extraction method did not perform well with certain datasets which had background objects that were in the similar intensity of the hand region. There is still scope for development for the presented method, especially in the preprocessing stage. The future work will be to try other background subtraction algorithms to achieve better performance even for smaller datasets.

Acknowledgment

I would like to thank Janarthanan, Purushothman, Priya and Aarav for spending their time and helping me create the custom dataset.

References

[1] C. Stauffer and W. E. L. Grimson, “Learning patterns of activity using real-time tracking,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 747–757, 2000.

[2] H. Khaled, S. G. Sayed, E. M. Saad, and H. Ali, “Hand Gesture Recognition Using Modified 1$ and Background Subtraction Algorithms,”Mathematical Problems in Engineering, 20-Apr-2015. [Online]. Available: https://www.hindawi.com/journals/mpe/2015/741068/. [Accessed: 08-Apr-2020].

[3] Murthy GR, Jadon RS. A review of vision based hand gestures recognition. International Journal of Information Technology and Knowledge Management. 2009 Jul;2(2):405-10.

[4] Thittaporn Ganokratanaa, and Suree Pumrin, “Hand Gesture Recognition Algorithm for Smart Cities based on Wireless Sensor,” International Journal of Online and Biomedical Engineering, vol. 13, no. 6, 2017.

[5] M. Yasen and S. Jusoh, “A systematic review on hand gesture recognition techniques, challenges and applications.”PeerJ Computer Science. 16-Sep-2019.

[6] S. Saha, “SparshaSaha/Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-Network,”GitHub, 24-Feb-2020. [Online]. Available: https://github.com/SparshaSaha/Hand-Gesture-Recognition-Using-Background-Elllimination-and-Convolution-Neural-Network. [Accessed: 09-Apr-2020].

[7] H. Park and K. Hyun, “Real-time hand gesture recognition for augmented screen using average background and camshift,” in Proceedings of the 19th Korea-Japan Joint Workshop on Frontiers of Computer Vision (FCV ’13), pp. 18–21, Incheon, Republic of Korea, January-February 2013.

[8] G. Ilango, “Hand Gesture Recognition using Python and OpenCV - Part 1,”Gogul Ilango, 06-Apr-2017. [Online]. Available: https://gogul.dev/software/hand-gesture-recognition-p1. [Accessed: 09-Apr-2020].

[9] “Motion Analysis and Object Tracking,”Motion Analysis and Object Tracking - OpenCV 3.0.0-dev documentation. [Online]. Available: https://docs.opencv.org/3.0-beta/modules/imgproc/doc/motion_analysis_and_object_tracking.html. [Accessed: 09-Apr-2020].

[10] Attwenger. A, ”Advantages and Drawbacks of Gesture-based Interaction”, Grin Publishing, 2017.

[11] C. Martin, ”Background subtraction using running gaussian average: a color channel comparison”, Seminar aus Bildverarbeitung und Mustererkennung, 2014.

[12] C. Stauffer and W. E. L. Grimson, “Adaptive background mixture models for real-time tracking,” in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR ’99), vol. 2, IEEE, Fort Collins, Colo, USA, June 1999.

[13] S. Suzuki and K. Be, “Topological structural analysis of digitized binary images by border following,” Computer Vision, Graphics and Image Processing, vol. 30, no. 1, pp. 32–46, 1985.

[14] Konstantinos G. Derpanis. “A Review of Vision-Based Hand Gestures”, 2004.

[15] Sushmita Mitra, and Tinku Acharya, “Gesture Recognition: A Survey”, IEEE Transactions on Systems, Man and Cybernetics–Part C: Applications and Reviews, 2007.

[16] Premaratne P. ”Historical development of hand gesture recognition”,Inhuman computer interaction using hand gestures, pp. 5-29, Springer, Singapore, 2014

[17] Weiguo Z, Congyi L, Xin J, Peng L, Haoyao C, Yun-Hui L, ”Real-time implementa-tion of vision-based unmarked static hand gesture recognition with neural networks based on FPGAs.”International conference on robotics and biomimetics, IEEE , 2017.

[18] Ananyaa S, Ayush K, Kavleen K, Shivani J, Richa U, Sameer P, ”Vision based static hand gesture recognition techniques.”, International conference on communication and signal processing, IEEE, 2017.

[19] A. Mavi, “Sign Language Digits Dataset,” Kaggle, 24-Dec-2017. [Online]. Available: https://www.kaggle.com/ardamavi/sign-language-digits-dataset. [Accessed: 17-Apr-2020].

[20] T. Mantecón, C.R. del Blanco, F. Jaureguizar, N. García, “Hand Gesture Recognition using Infrared Imagery Provided by Leap Motion Controller”, Int. Conf. on Advanced Concepts for Intelligent Vision Systems, ACIVS 2016, Lecce, Italy, pp. 47-57, 24-27 Oct. 2016. (doi: 10.1007/978-3-319-48680-2_5)

[21] T. Mantecón, C.R. del Blanco, F. Jaureguizar, N. García, “Hand Gesture Recognition using Infrared Imagery Provided by Leap Motion Controller”, Int. Conf. on Advanced Concepts for Intelligent Vision Systems, ACIVS 2016, Lecce, Italy, pp. 47-57, 24-27 Oct. 2016. (doi: 10.1007/978-3-319-48680-2_5)