HAND GESTURE RECOGNITION USING TRANSFER LEARNING

Group - 13

University of Calgary

ABSTRACT

Using hand gestures as a means to interact with a computer is one of the most easiest and natural way of human-computer interaction. It proves to be of great aid to the hearing and speech impaired people by implementing the concept to automate sign language recognition. It can also be used in the process of preserving the cultural heritage - by automating the hand gestures used in classical and folk dances. In this project, an attempt is made to create a gaming application that allows humans to play the game of rock, paper, scissor with a computer. Data for the same was collected using webcam and using transfer learning via MobileNet, the model was trained. Training accuracy of 100% and validation accuracy of 99% was achieved.

***Index Terms*—** Transfer Learning, Data Augmentation, MobileNet, Region of Interest (ROI), Convolutional Neural Network.

1. INTRODUCTION

With the advent of Virtual Reality (VR) and Machine Learning (ML), hand gesture recognition (HGR) has become quite popular. HGR is not just limited to human-computer interaction, but also includes areas like pattern recognition, signal processing and computer vision. But since the communication of hearing impaired people is primarily by the means of actions involving hand movements, HGR is the topic under focus for this paper.

Apart from sign language recognition, HGR also has application in other areas. For instance [1] uses convolutional neural networks (CNN) to learn images of hand gestures associated with traditional dance moves, optimizes the hyperparameters based on the effectiveness of CNN and attempts to increase the accuracy by using double transfer learning.

Other notable applications of HGR includes intelligent driving [2] wherein the vehicle navigation and information entertainment system can reduce the distraction caused by the driver with the aid of gesture recognition system. It can also be used for robot communication.

HGR can also be used in applications employing gesture control interface and one such instance is the game of rock, paper, scissor. So this paper attempts on learning various images of hands showing the sign of rock, paper and scissor. Transfer learning is used for training the model, alternatively VGG16 is also used to train the model and finally both the approaches are compared. All the codes used for - collection of data and preparation of the dataset, training the model, plotting the results from both the approaches and the auxiliary images used by the bot (computer) to signal rock, paper and scissor are present in the following github repository: `https://github.com/mananpatel126/ENEL-645-Group-13/tree/main/Project`.

Thanks to Dr. Roberto for providing the opportunity to work on the project.

2. RELATED WORK

With the tremendous progress in the field of deep learning, HGR has grown to be popular recently. Ozcan and Basturk in [3], proposed transfer learning (using AlexNet) based CNN for HGR. In the history of HGR for the first time, they optimized the hyperparameters using optimization algorithms like Artificial Bee Colony (ABC), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Based on the result of comparison of the above algorithms, they proceeded with ABC algorithm to tune CNN for sign language digit dataset and also for the Thomas Moeslund's gesture recognition dataset.

In [4], the recognized six static as well as eight dynamic hand gestures are used to command the computer to perform certain tasks. For the purpose of hand shape recognition, they used CNN based transfer learning approach. And VGG16 being the pretrained model used for transfer learning.

Khalil and Mostefa in [2] worked on the American Sign Language (ASL) dataset and proposed a ConvNet model (based on CNN) for hand gesture recognition and optimizing the performance of the model by using TensorBoard (which is a tool that provides measurements and visualization required by machine learning workflows). They also compared the results with transfer learning based models trained using different versions of MobileNet. And concluded with MobileNetV2 being the model yielding higher accuracy with lower latency compared to MobileNetV1.

Mukul and Sherly in [5], worked on recognization of static as well as dynamic hand gestures. For detecting the gesture, double channel CNN is used over a small dataset to detect the presence of the gesture and as soon as the gesture is

Mode	Paper	Rock	Scissor	Wait	Total
Train	2000	2000	2000	2000	8000
Validation	310	310	310	310	1240
Test	310	310	310	310	1240
Total	2620	2620	2620	2620	10480

Table 1. Segregation of Collected dataset

detected, skeletal based approach along with transfer learning are used to classify the gesture.

Cote-Allard in [6], proposed a transfer learning supported CNN to capture more general and robust features on surface electromyography (sEMG) gesture recognition data acquired by the author.

3. METHODOLOGY

This paper can broadly be classified into data collection, data preparation, training and testing/application. Each of which are discussed below:

3.1. Data Collection

The target of the paper being implementation of the game of rock, paper and scissor, images of all the three as well as ”wait”(containing no gesture or potentially empty frame) are collected using webcam and OpenCV. Figure-1 shows all the four classes involved in the dataset. Table-1 portrays the the number of images in each class as well as the number of images in train, validation and test cases. So 2620 images of each class are collected which collectively totals to 10480 images in the dataset.

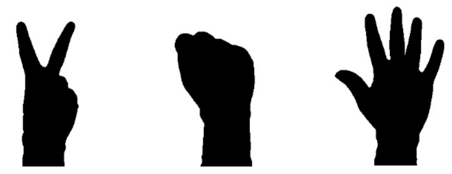


Fig. 1. Sample images of all four classes

Not the entire image captured by the webcam is used since it increases the image size and unnecessary computation time. So a Region of Interest (ROI) is defined to form a window of 224x224 i.e. the image captured is defined to be of the shape (224, 224). Furthermore, the three channel image captured is converted to gray scale so that while testing the model, skin colour would not have any negative impact on the result.

Now one might question - how would the object be differentiated from the background? So as a solution to this problem, accumulated weight is calculated and a threshold value

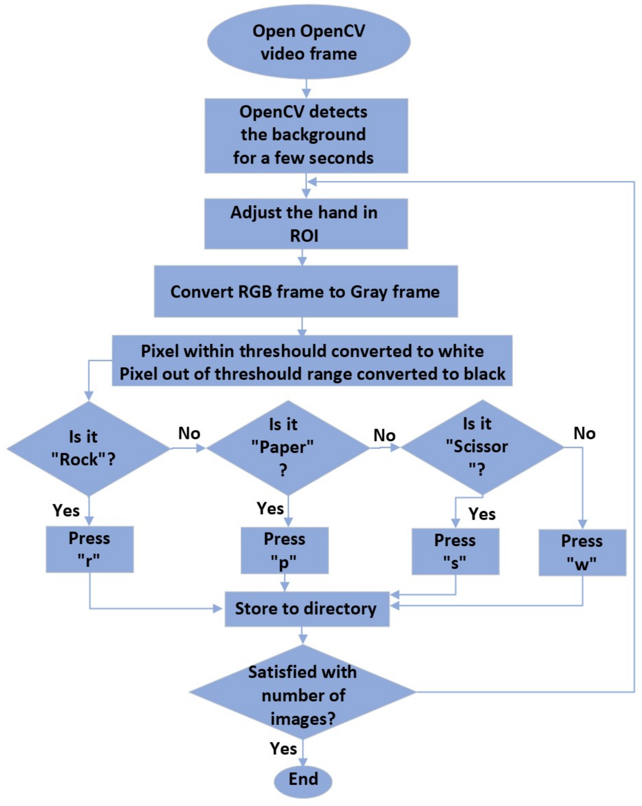

Open OpenCV video frame
OpenCV detects the background for a few seconds
Adjust the hand in ROI
Convert RGB frame to Gray frame
Pixel within threshould converted to white
Pixel out of threshould range converted to black
Is it "Rock"?
No
Is it "Paper" ?
No
Is it "Scissor "?
No
Yes
Press "r"
Yes
Press "p"
Yes
Press "s"
Press "w"
Store to directory
Satisfied with number of images?
Yes
End

Fig. 2. Flowchart for data collection

is applied which enables to differentiate the image pixel of the object from the background.

Once the required image is captured, it is manually stored in the associated directory. For instance if the image of ”paper” is captured, then by pressing the key ”p” store that image in the folder containing all paper images, and likewise for other classes as well. Figure-2 shows the entire process carried out to create the dataset.

The entire process of data collection is carried out using Jupyter Notebook since OpenCV is not supported on Google Colab.

3.2. Data Preparation

It is known that greater the amount of data accessed by the machine learning model, better is the performance of the model. And since limited data is collected in the collection step, data augmentation is used which allows the model access to more training data than the previously collected original data. Data augmentation (as used in [7]) basically increases the training data by performing little computation on the base data.

Hence before training the model, following data augmentations were considered: width shift range of 0.1 - which implies that the image is allowed to be randomly shifted horizon-

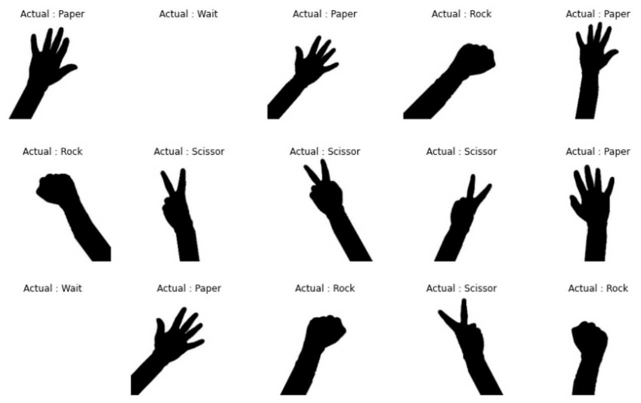


Fig. 3. Random batch of augmented data

tally by -10px to 10px, height shift range of 0.1 - i.e. image allowed to be shifted vertically by -10px to 10px, horizontal flip set to ”True”, shear range set to 0.1 so as to allow distortion of the image along an axis, rotation range set to 45 which allows the data to be rotated till 45°and zoom range so as to zoom in and zoom out the image by 50% of the original. All these effects can be seen in the Figure-3 showing a randomly selected batch of data after applying data augmentation.

3.3. Training

Once the process of data augmentation is complete, the data is shuffled and split into three parts namely for training, validation and testing as shown in the Table-1. Shuffling is done to address the presence of bias in the data while training. After that, train, validation and test batches are created. So that during each epoch, the data in batches would collectively be processed.

MobileNet is used to load model pre-trained on Imagenet dataset. But the base model is not allowed to train at first so as to freeze the initial layers while training. So keeping those layers frozen, the flatten and two dense layers are added at the end. Rectified Linear Unit (ReLU) and softmax activation are used in the respective dense layers.

While compiling the model, Stochastic Gradient Descent (SGD) optimizer is used setting the learning rate to be 0.001. Then the condition is set to gradually lower the learning rate - validation loss is tracked and if the same error occurs twice, the learning rate is reduced by 80%. The minimum learning rate is defined to be 0.00005 and the early stopping criterion is set to patience level 2. Using all these criterion, the model is trained for 10 epochs. The model is saved and the plots of error vs epoch and accuracy vs epoch are plotted is shown in section-4.

Post this, the model is allowed to train for fine tuning. Using the SGD optimizer and setting all the criterion of learning rate and early stopping in a similar fashion as done prior to fine tuning, the model is trained for a couple of epochs. This new model is then saved and loaded again to plot the results as shown in section-4.

The reasons behind using MobileNet to train the model are as follows: MobileNet is a class of CNN available openly, it has 28 layers in its architecture so it belongs to a small class of model, it offers low latency and low power consumption.

Apart from that, for the purpose of training the model, Google Colab and the GPU offered by the same were used.

3.4. Application

This paper uses the application of HGR for the purpose of playing the game of rock, paper and scissors. First of all, the model prepared in section-3.3 is loaded. Region of Interest is defined and threshold is set so as to differentiate the object (hand) from the background in a manner as shown earlier in section-3.1.

The moves of the bot are randomized i.e. while the game is on, the computer will randomly select its action of rock, paper or scissor. Then the rules of the game are set so as to decide the winner of the game. For instance if the bot’s move is ”paper” and the humans move is ”rock”, then since the paper covers rock, the bot is declared to be the winner of that particular round. Based on the combinations of humans and bot’s move, the result can be winner, loser and tie. Using a count variable, the cumulative sum of number of wins for each of the entity is kept to keep track of total wins throughout all the rounds.

Once the video is allowed to be captured, for the first few seconds it detects the background and then the game begins. The scores of the game are updated as per the set rules of the game. The snapshots of the game are shown in section-4.

4. RESULTS

4.1. Model Results

As shown in Figure-4, the training loss start from 25% which drastically drops to about 3.5% at the end of first epoch and then gradually saturates at 0.5%. On the other hand validation loss starts at 7%, follows (parallel to) the plot of training loss and saturates at about 2%. Figure-5 shows the plot of training and validation accuracy to the number of epochs. Training accuracy starts from 95.5% and shoots up to 100% by the end of first epoch while the validation accuracy fluctuates between 99% and 100%.

From Figure-6, it can be drawn that with the aid of fine tuning, the training loss starts from 0.4% and ends up at just about 0.45% by the conclusion of first epoch while the validation loss decreases from 1.7% to 1.55%. The Figure-7 shows that the training accuracy in fine tuning stays 100% while the validation accuracy starts at 99.75% and drops to 99.6% at the end of first epoch.

After loading the transfer learning model, the confusion matrix showing the summary of prediction result of a test

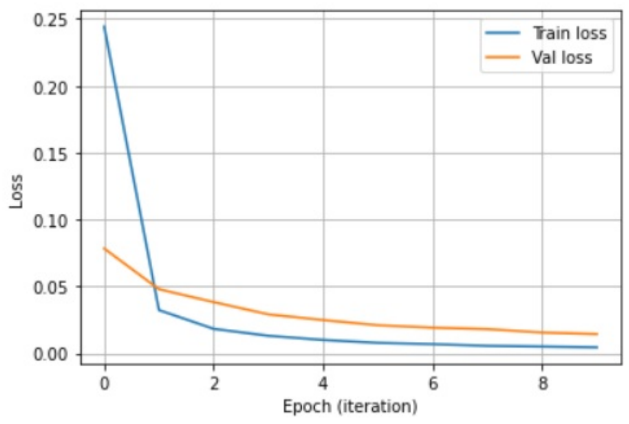


Fig. 4. Error Vs Number of Epoch for training and validation in Transfer Learning

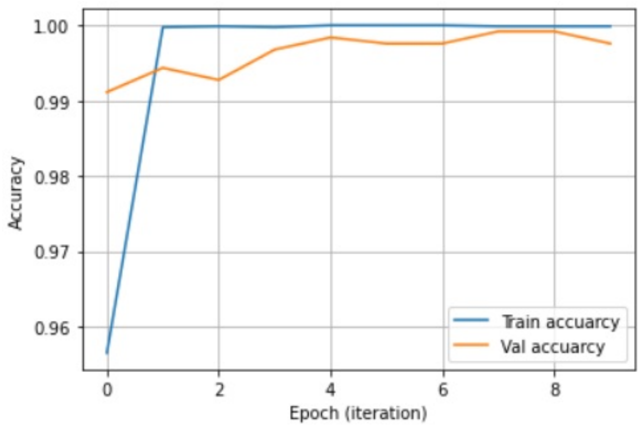


Fig. 5. Accuracy Vs Number of Epoch for training and validation in Transfer Learning

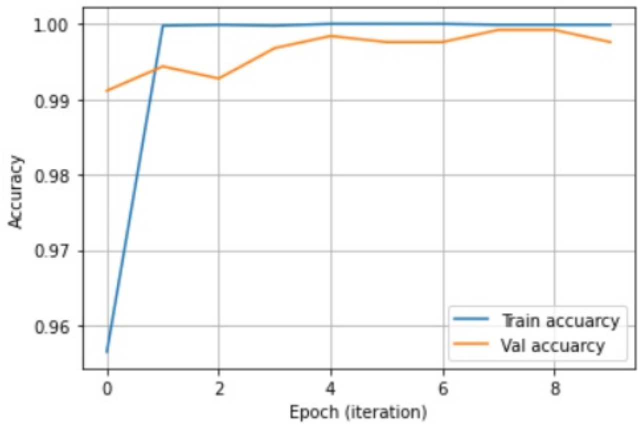


Fig. 6. Loss Vs Number of Epoch for training and validation after fine tuning

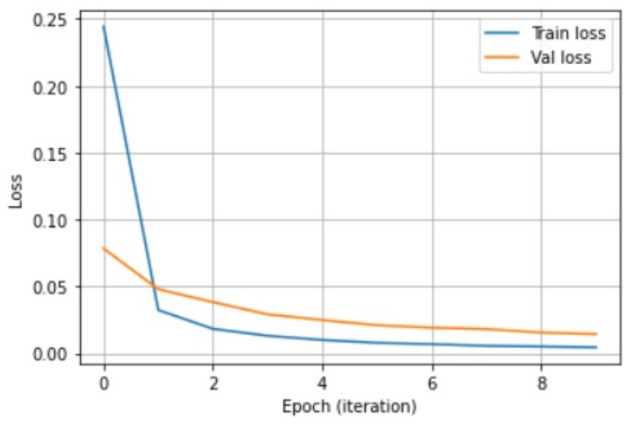
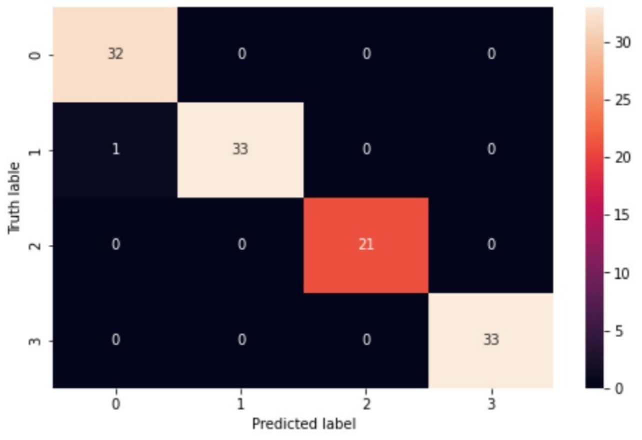


Fig. 7. Error Vs Number of Epoch for training and validation after fine tuning



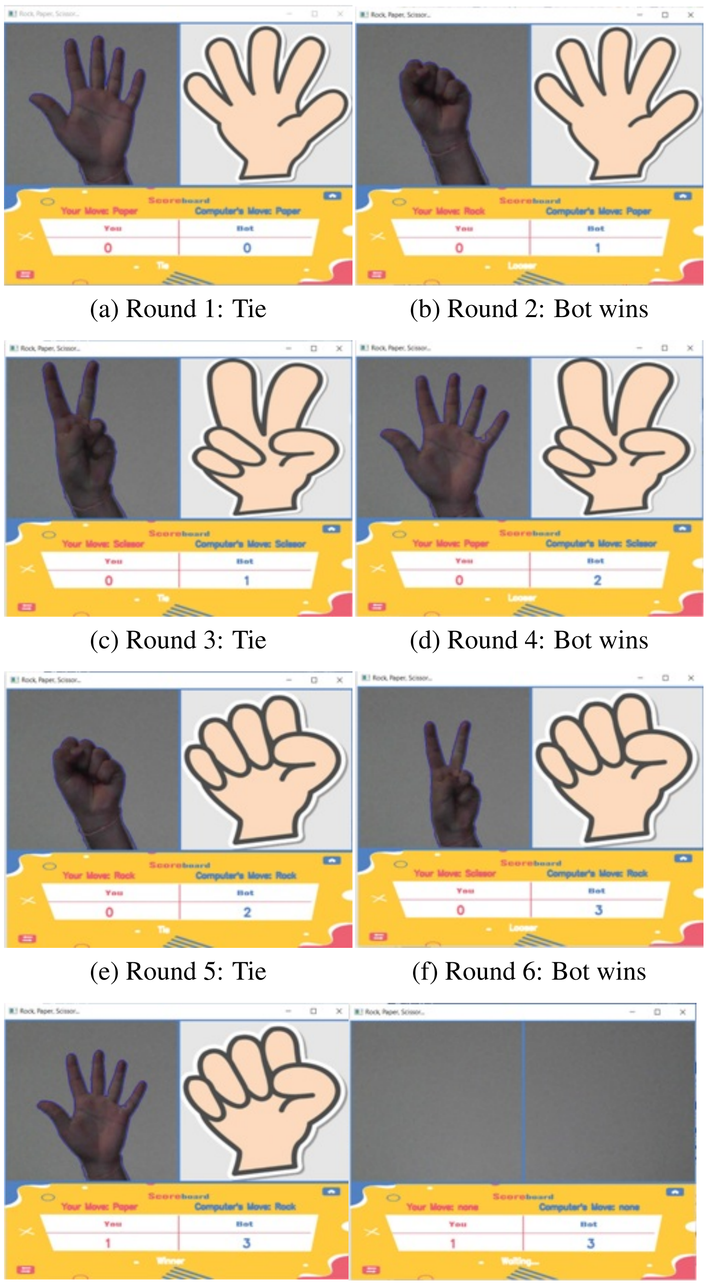
	0	1	2	3
0	32	0	0	0
1	1	33	0	0
2	0	0	21	0
3	0	0	0	33

Fig. 8. Confusion Matrix

batch is shown in Figure-8. The marking of ”0”, ”1”, ”2” and ”3” on X and Y axis represents ”Paper”, ”Rock”, ”Scissor” and ”Wait” respectively. Hence it can be made out that there was just one case of wrongly predicted instance - one of the originally ”Rock” image was predicted to be ”Paper”. In all other cases, the predicted class exactly matches the actual class of the data.

4.2. Game Results

The interface of the game is shown in Figure-9. With each round, the scoreboard is updated. It shows the action of human, action of the bot, number of wins for human, number of wins for bot and the outcome of the round in the perspective of human i.e. if human wins, it produces ”winner”; if human loses, it shows ”loser” and ”tie” if the action of human and bot matches.



(a) Round 1: Tie (b) Round 2: Bot wins

(c) Round 3: Tie (d) Round 4: Bot wins

(e) Round 5: Tie (f) Round 6: Bot wins

(g) Round 7: Human wins (h) Round 8: No result

Fig. 9. Sequential snapshots of the game

5. CONCLUSION

6. REFERENCES

[1] Anuja P. Parameshwaran, Heta P. Desai, Rajshekhar Sunderraman, and Michael Weeks, "Transfer learning for classifying single hand gestures on comprehensive bharatanatyam mudra dataset," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2019, pp. 508–510.

[2] Khalil Bousbai and Mostefa Merah, "A comparative study of hand gestures recognition based on mobilenetv2 and convnet models," in *2019 6th International Conference on Image and Signal Processing and their Applications (ISPA)*, 2019, pp. 1–6.

[3] Tayyip Ozcan and Alper Basturk, "Transfer learning-based convolutional neural networks with heuristic optimization for hand gesture recognition," *Neural Computing and Applications*, vol. 31, no. 12, pp. 8955–8970, 2019.

[4] Soeb Hussain, Rupal Saxena, Xie Han, Jameel Ahmed Khan, and Hyunchul Shin, "Hand gesture recognition using deep learning," in *2017 International SoC Design Conference (ISOCC)*, 2017, pp. 48–49.

[5] Mukul Nair and Sherly Noel, "Hand gesture recognition using double cnn and transfer learning," in *2021 International Conference on Communication, Control and Information Sciences (ICCISc)*, 2021, vol. 1, pp. 1–6.

[6] Ulysse Côté-Allard, Cheikh Latyr Fall, Alexandre Campeau-Lecours, Clément Gosselin, François Laviolette, and Benoit Gosselin, "Transfer learning for semg hand gestures recognition using convolutional neural networks," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, 2017, pp. 1663–1668.

[7] Ali A Alani, Georgina Cosma, Aboozar Taherkhani, and TM McGinnity, "Hand gesture recognition using an adapted convolutional neural network with data augmentation," in *2018 4th International conference on information management (ICIM)*. IEEE, 2018, pp. 5–12.