# ROCK, PAPER, SCISSOR GAME APPLICATION WITH HAND GESTURE RECOGNITION USING OPENCV AND 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 OpenCV and using different transfer learning pre-trained models i.e. MobileNet, VGG16 and ResNet50, training accuracy of 100% and validation accuracy of 100% was achieved.

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

## 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.

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

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 through VGG16, MobileNet and ResNet50 models and finally all the approaches are compared. All the codes used for - collection of data and preparation of the dataset, training the model, plotting the results from all the approaches and playing the Rock, Paper, Scissor game with the bot (computer) are present in the following github repository: `https://github.com/mananpatel126/ENEL-645-Group-13/tree/main/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 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 segmentation, 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 OpenCV library. 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.

(a) Wait (b) Scissor (c) Rock (d) Paper

**Fig. 1**. Sample images of all four classes

| 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

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 capture the image-frame and it converted to the shape of (224, 224) which is the required input size of transfer learning model. 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 as well to increase the ease of object detection.

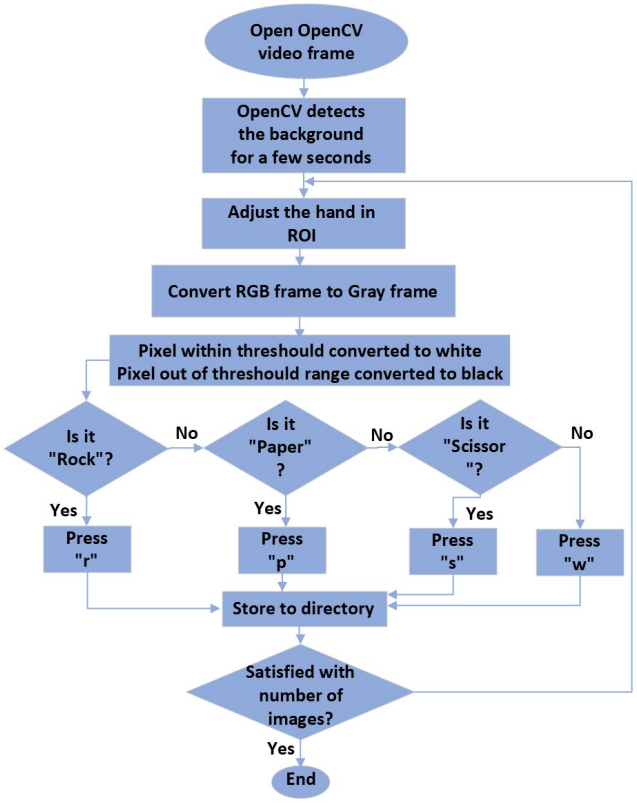



**Fig. 2**. Flowchart for data collection

Now one might question - how would the object be differentiated from the background? So to distinguish between the background and the object, the background's cumulative weighted average is calculated and deducted it from the frame that has any object in front of the background that can be identified as object(hand). Calculated the accumulated weight for various frames (here for 70 frames) and the accumulated avg for the backdrop accomplishes this. Then the cumulative average for the background is deducted from every frame read after 70 frames to detect any object that covers the backdrop [7].

Thereafter, utilizing the calculated threshold value, the contour is determined using cv2.findContour which returns the outermost contour. With the help of the contours, the presence of object(hand) in the ROI is identified.

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.

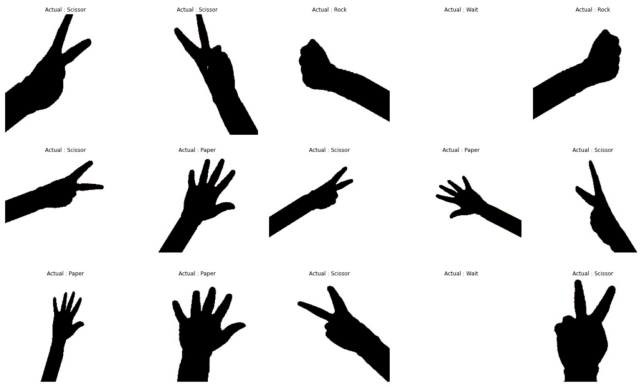


**Fig. 3**. Random batch of augmented data

### 3.2. Data Augmentation

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 [8]) basically increases the training data by performing little computation on the base data.

To train the model, random generated batches of tensor image data with real time data augmentation is used with the help of ImageDataGenerator of Keras. Following data augmentation is applied: width shift range of ±10%, height shift range of ±10%, horizontal flip, shear range set to ±10% so as to allow distortion of the image along an axis, rotation range set to 75°which allows the data to be rotated till ±75°and zoom range so as to zoom in by 20% and zoom out the image by 200% of the original. All these effects can be seen in the Figure-3 showing a randomly selected batch of data after applying data augmentation. For all the three transfer learning models, identical batch size chosen to be 64.

### 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.

Transfer learning is the process of transferring or repurposing learnt features from a pre-trained neural network to a new dataset. The three approaches used are discussed as under:

#### *3.3.1. VGG16*

VGG16 is a convolutional neural network trained on a portion of ImageNet dataset which contains more than 14 million images divided into 22,000 categories [9]. The model of VGG16 consists of 16 convolutional and max pooling layers, 3 dense layers for fully convolutional layers resulting into 138 million parameters [10], [11]. The same pre-trained model is used to train the model.

Since the entirely pre-trained model cannot be used to train the four-class classification model, pre-trained VGG16 model is imported with the full convolutional layers cut off which can also be referred to as ”top” model. Fine tuning approach is employed to train the model wherein the process starts with computing a new ”top” portion of the model, followed by freezing the pre-trained convolution layers and finally un-freezing the last few pre-trained layer. ADAM optimizer is used while compiling the model.

#### *3.3.2. Mobilenet*

MobileNet is pre-trained on Imagenet dataset and consists of 4.2 million parameters. In the initial training procedure, keeping the initial layers frozen, only the Fully Convolutional Layers are trained. While compiling the model, Stochastic Gradient Descent (SGD) optimizer is used, also softmax is used as the activation layer in the output layer.

Merits of using MobileNet pre-trained model are as follows: MobileNet is nearly as accurate as VGG16 while being 32 times smaller and 27 times faster [12]; 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.

#### *3.3.3. ResNet*

ResNet [13] has 24 million parameters however trainable parameters are 1 million. Adamx optimizer used. ResNet model has lower filters and lower complexity as compare to VGG nets.

For all the three models, learning rate is kept to be 0.001. Then the condition is set to gradually lower the learning rate - validation loss is tracked and if the same loss occurs twice, the learning rate is reduced by 50%. 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, initially the model is trained for 20 epochs and fine tuned for 5 epochs. After each epoch, the best fit model is saved and the plots of error vs epoch and accuracy vs epoch are plotted as shown in section-4. Default input image shape is (224, 224, 3) for all the three transfer learning models.

For the purpose of training the model, Jupyter Notebook, 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

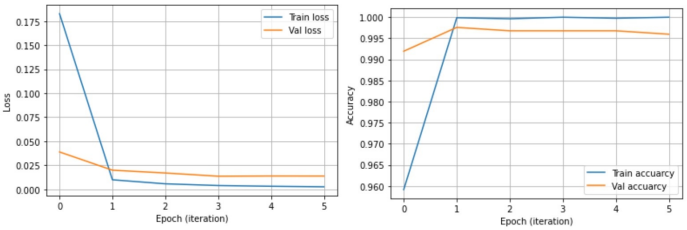
As shown is Table-2, the computational time for MobileNet model is much lower than other two model, keeping almost same accuracy. The reason being - trainable parameters 501,814 out of total parameters 3,730,678 in MobileNet model. However, in case of ResNet model, there are 1,003,574 trainable parameters out of total 24,591,286 parameters which effectively increases the training time by four times as compared to MobileNet model.

| Model | MobileNet | VGG16 | ResNet |
|---|---|---|---|
| Initial Model Train Time (min) | 18.41 | 116.66 | 86.21 |
| Fine Tuned Model Time (min) | 36.13 | 198.38 | 137 |
| Top-1 accuracy (%) | 99.19 | 99.19 | 99.27 |

**Table 2**. Comparison of model training times and accuracy

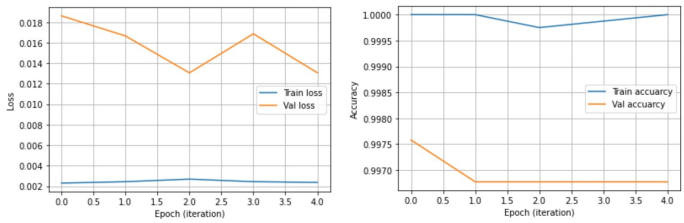
The plots of Loss Vs Number of Epochs and Accuracy Vs Number of Epochs are as shown in Figures-4, 5, 6, 7, 8 and 9.

The confusion matrix shown in Figure-10, illustrates the summary of prediction result a random test batch. The diagonal elements in the matrix indicates accurate prediction i.e. both the actual and predicted label are same. Since all the transfer learning models give similar accuracy on test data, only one confusion matrix has been plotted.



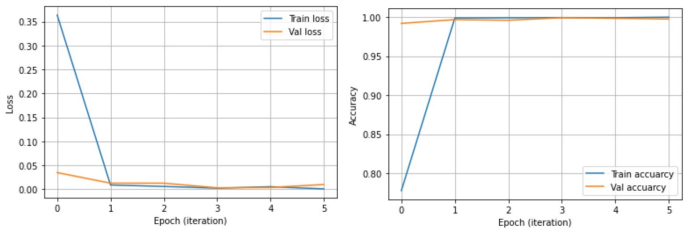

(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 4**. Initial training results for MobileNet model



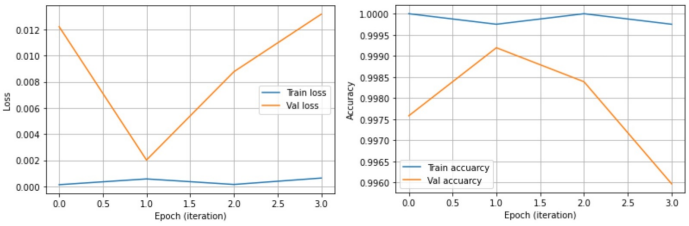

(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 5**. Fine tuning results for MobileNet model




(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 6**. Initial training results for VGG16 model

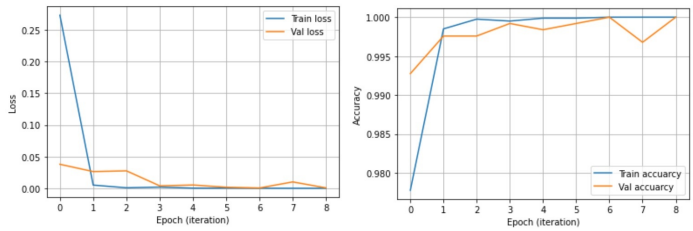



(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 7**. Fine tuning results for VGG16 model

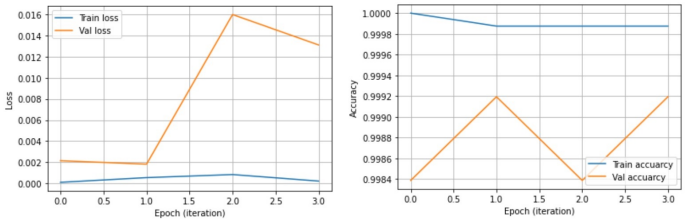
### 4.2. Game Results

The interface of the game is shown in Figure-11. With each round the scoreboard is updated. It shows the action of user, action of bot, number of wins for user, number of wins for bot and the outcome of each round in the perspective of user




(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 8**. Initial training results for ResNet model




(a) Loss Vs Epoch (b) Accuracy Vs Epoch

**Fig. 9**. Fine tuning results for ResNet model

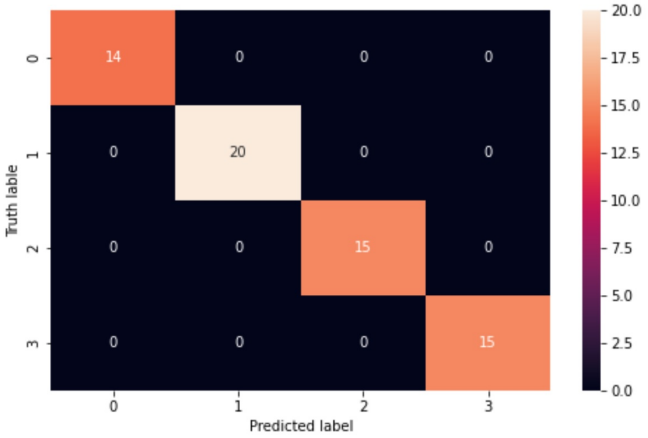
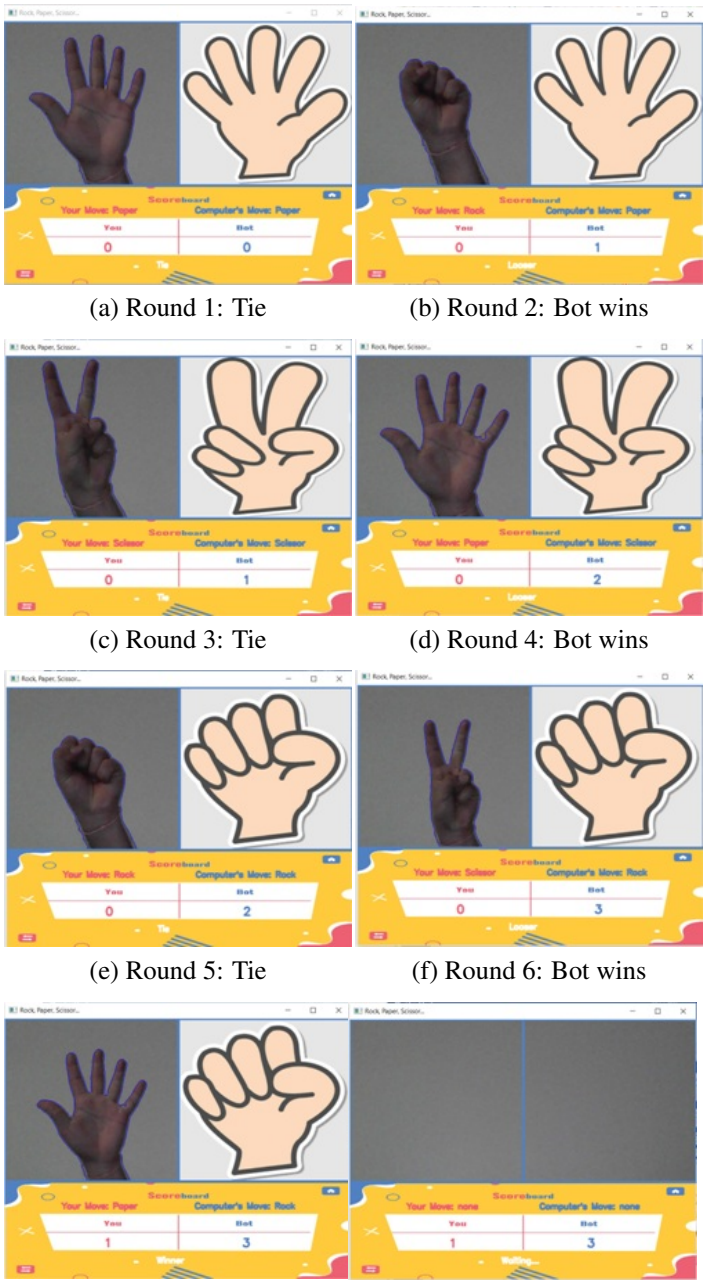



**Fig. 10**. Confusion Matrix

i.e. if user wins, it shows ”Winner”; if user loses, it shows ”looser” and ”tie” if the action of user and the 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. 11**. Sequential snapshots of the game

## 5. CONCLUSION

To reiterate, VGG model gives the best accuracy in predicting the live model using OpenCV, although is six times slower compared to MobileNet. Moreover, the size of VGG model is 171 megabytes which is approximately 12 times larger in size than MobileNet i.e. 14.3 megabytes. Hence, MobileNets typically are not as accurate as these other large, resource-heavy model but they still actually perform very well, with a relatively small reduction in accuracy. On the other hand, ResNet gives comparatively better accuracy than MobileNet. Nevertheless, the time complexity and the size of the model on the disk of ResNet is 4 times and 20 times higher that MobileNet model respectively.

Based on the above analysis, it can be concluded that MobileNet model achieves comparable results to other models with much lower computational complexity and model size.

## 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] DataFlair, "Sign language recognition using python and opencv," `https://data-flair.training/blogs/sign-\\language-recognition-python-ml\\-opencv/`, 2022.

[8] 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.

[9] Deeplizard, "Mobilenet image classification with tensorflow's keras api," `https://deeplizard.com/learn/video/OO4HD-1wRN8`, 2022.

[10] Karen Simonyan and Andrew Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] LearnDataSci, "Hands-on transfer learning with keras and the vgg16 model," `https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/`, 2022.

[12] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.