Hand Gesture Recognition Using Faster R-CNN Inception V2 Model

Conference Paper · July 2019

DOI: 10.1145/3352593.3352613

2 authors:

Rubin Bose S.
Madras Institute of Technology
2 PUBLICATIONS **1** CITATION
SEE PROFILE

Sathiesh Kumar
Anna University, Chennai
29 PUBLICATIONS **96** CITATIONS
SEE PROFILE

Hand Gesture Recognition Using Faster R-CNN Inception V2 Model

Rubin Bose S
Department of Electronics Engineering
MIT Campus /Anna University
Chennai Tamil Nadu India
rublins@gmail.com

Sathiesh Kumar V
Department of Electronics Engineering
MIT Campus /Anna University
Chennai Tamil Nadu India
sathieshkumar@annauniv.edu

ABSTRACT

The realtime hand motion recognition under unconstrained environment is a challenging computer vision problem. The change in illumination and non-uniform background condition makes it very difficult to perform real-time hand gesture recognition operations. This paper demonstrates a region-based convolutional neural network for real-time hand gesture recognition. The custom dataset is captured under unconstrained environments. The Faster region-based convolutional neural network (Faster-RCNN) with Inception V2 architecture is used to extract the features from the proposed region. The average precision, average recall, and F1-score are analyzed by training the model with a learning rate of 0.0002 for Adaptive Moment Estimation (ADAM) and Momentum optimizer, 0.004 for RMSprop optimizer. The ADAM optimization algorithm resulted in better precision, recall and F1-score values after evaluating custom test data. For ADAM optimizer with intersection over union (IoU) =0.5:0.95, the observed average precision is 0.794, average recall is 0.833, and the F1-score is 0.813. For an IoU of 0.5, ADAM optimizer resulted in 0.991 average precision with a prediction time of 137ms.

CCS CONCEPTS

Computing methodologies, Computer vision problems, Object detection, Object recognition.

KEYWORDS

Region proposal, Faster-RCNN, Convolutional Neural Network, Hand gesture recognition, Inception-V2.

1 INTRODUCTION

Hand Gesture based Human-machine interaction plays an important role in interacting with machines. Numerous applications are being developed using gestures. Some of them include controlling robotics, video surveillance, multimedia video retrieval, etc.[1]. Performing a realtime gesture-based machine interaction under unconstrained environment is a challenging task. The challenges in tracking systems are self-occlusions, rapid motion, high degrees of freedom (DOF), processing speed and uncertain environments. Most of this problem is very common for tracking systems, but exclusively for hand recognition system, the major challenges are rapid motions and high DOF. The hand incorporates 27 bones ahead of with several sets of muscles and tendons. Normally, 27 DOF for one hand and 54 DOF for two hands [2]. Hand gesture recognition is based on two approaches: static gesture and dynamic gesture. Static gestures have a particular meaning for every static form of hand poses. Dynamic gestures are a sequence of static postures organized to form a single gesture within a time period [3].

In recent years, the evolution of Neural Network (NN) has resulted in the development of numerous deep learning algorithm through a special type of network called Convolutional Neural Networks (CNN). The position of the object in visual data is determined using NN algorithms, based on CNN's are called Object Detectors. Region Proposal Convolutional Neural Network (R-CNN) algorithm is developed and it is further refined as the Fast R-CNN [4]. Because, the time required to train, to detect and classify the objects are much faster than R-CNN. Fast R-CNN provides faster end-to-end training but still not suitable for in-situ object detection. The Faster R-CNN [5] is more suitable for in-situ applications that use a compact network and it requires a very short time to perform the detection with better state-of-the-art accuracy [6].

2 RELATED WORKS

Jonathan et al. [6] proposed an object detector by the integrated implementation of meta-architectures incorporated with Faster R-CNN. The author trained the model end-to-end on a distributed cluster using asynchronous gradient updates for Faster R-CNN. The network is trained and validated on the COCO dataset (8000 samples).

Wu et al. [7] proposed a Deep Dynamic Neural Network for hand gesture recognition on multimodal data incorporating RGB-D information and skeleton features. The author utilized the deep neural network to extract pertinent information from the data. This model integrates two different feature-learning techniques. Deep belief networks for processing of skeleton features and 3-D Convolutional Neural Networks for RGB-D data. Authors evaluated the model on the ChaLearn LAP dataset.

Javier et al. [8] implemented a Region-based Convolutional Neural Network for recognition and localization of hand gestures. The author trained and validated the neural network for two classes of hand gestures in a dynamic background and achieved a validation accuracy of 99.4% in gesture recognition and an average accuracy of 25% in ROI localization.

Shaoqing et al. [9] presented a Region Proposal Networks (RPN) for efficient and accurate generation of the proposed region. A unified, deep-learning-based object detection system called Faster

AIR '19, July 2–6, 2019, Chennai, India

ACM ISBN 978-1-4503-6650-2/19/07…$15.00
https://doi.org/10.1145/3352593.3352613

R-CNN incorporates two modules. A deep fully convolutional neural network proposes regions and a Fast R-CNN detector utilizes the proposed regions. This region proposal network enables deep object detection system to run at realtime frame rates. The quality of region proposal and the overall object detection accuracy is improved by learned RPNs.

Hsien et al. [10] proposed a convolution neural network (CNN) technique to recognize hand gestures. The author utilized the orientation, the skin model and the calibration of hand positions to obtain the training and testing data for the CNN. The author adopts a Gaussian Mixture Model (GMM) to train the skin model, which robustly filter out non-skin colors in an image. The preprocessed images are used to train the CNN. The validation of the method of recognizing human hand gestures gives a robust result for several hand poses, orientations and lighting conditions.

Ibrahim et al. [11] proposed a deep learning algorithm to classify ten different hand gestures from the impedance variation of two monopole antennas. The impedance is converted into grayscale spectrogram images and classified using a deep convolutional neural network. The author implemented this method to control the operations of electronic gadgets.

Chaudhary et al. [12] implemented an Artificial Neural Network (ANN) for a light invariant hand gesture recognition system. The orientation histogram is used to extract the unique feature vector of a hand gesture and it is compared using supervised ANN. The system is trained in a manner such that six different gestures are being recognized. The audio file of the corresponding matched gesture is played during recognition. For an extreme light intensity varying environments, the overall accuracy of 92.86% is achieved.

Based on the extensive literature survey, it has been identified that the accuracy in realtime detection of hand gestures could be further improved. In this paper, the Faster R-CNN Inception V2 model with gradient descent optimization is used to perform real-time hand gesture recognition under unconstrained environment.

3 METHODOLOGY

3.1 Convolutional Neural Network

Image recognition and classification are more effective by using Convolutional Neural Network (CNN) Architectures. Fig. 1 show the architecture of CNN. It consists of an input layer, convolution layer, pooling layer, fully connected layer, and an output layer. CNN's are also used in object detection tasks. The three generations of CNNs are region-based CNN (R-CNN), Fast region-based CNN (Fast R-CNN) [4] and Faster region-based CNN (Faster R-CNN). In this paper, the proposed CNN architecture for hand gesture recognition is Faster R-CNN [5] with Inception V2 model [13].

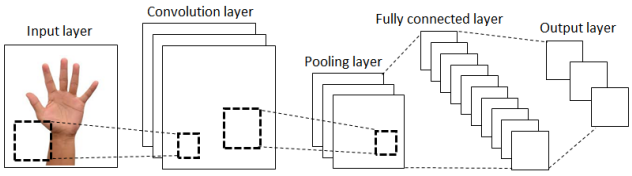

Figure 1: Block diagram of CNN

3.1.1 Faster R-CNN. Figure 2 shows the functional block diagram of Faster R-CNN. Faster R-CNN is the modified version of Fast R-CNN, which adds a Region Proposal Network (RPN) for generating object proposals to Fast R-CNN. To generate region proposals in R-CNN and Fast R-CNN, external object proposal modules such as Edge Boxes or Selective Search are utilized. From the last convolutional layer of CNN, the RPN reuses the feature maps for the generation of the proposed region. In Faster R-CNN, both region proposals and classification takes place in a single network.

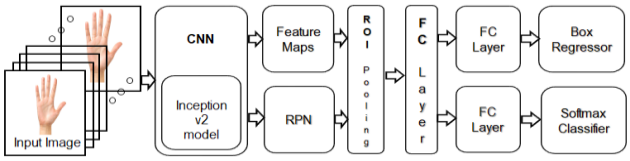

Figure 2: Block diagram of Faster R-CNN.

In recent works, the generation of box proposals has been carried out using a neural network. Which typically is a collection of boxes enwrap on the image at different aspect ratio, scale, and spatial location are called anchors. It is also called priors or default boxes. For each anchor box, a model is trained for two predictions. First, for each anchor box, a discrete class prediction is carried out. Second, a stable prediction with an offset is carried out through which the anchor box needs to be transited to fit the ground truth bounding box [12]. The concept of anchor box minimizes classification and regression losses. The best suitable ground truth box b is identified for each anchor a. If it matches then a is called as a positive anchor and it is assigned a class label $y_a \in \{1 \ldots . K\}$. Secondly, a vector encoding of box b with respect to anchor a is verified. $\Phi(b_a; a)$ is called an encoding box. If it does not match, a is called as a negative anchor and the class label is to be $y_a = 0$. For the anchor a, the predicted box encoding is $floc(I; a; \theta)$ and its equivalent class is $fcls(I; a; \theta)$, where I is the image and θ is the model parameters. Then the weighted sum of a location-based loss and a classification loss is measured as the loss for anchor a [4].

$$L(a, I, \theta) = \alpha.1[a \text{ is positive}].\ell_{loc}\left(\phi(b_a; a) - f_{loc}(I; a; \theta)\right) + \beta\ell_{cls}\left(y_a, f_{cls}(I; a; \theta)\right) \quad \ldots (1)$$

Where α and β are the weight balancing localization and classification losses. Equation 1 is averaged for various anchors and reduced with respect to parameters to train a model. Anchors have significant associations with accuracy and computation.

The spatially local correlation is investigated by a convolutional layer for its inputs and mapped into the next layer as a feature map. For different kernels, different feature maps are generated. The activity of the i^{th} feature map in the l^{th} layer is calculated by equation 2.

$$y_i^l = \sum_j f(w_{i,j}^l * y_j^{l-1} + b_i^l) \quad \ldots .. (2)$$

Where, y_i^l is the i^{th} feature map in l^{th} layer. $w_{i,j}^l$ is the convolutional kernel (weight matrix) for y_j^{l-1}, and the weight matrix connects y_j^{l-1} to the feature map y_i^l, b_i^l is a bias of the i^{th} feature map in the l^{th} layer, and f (.) is a nonlinear activation function, such as the rectified linear units (ReLU) function or sigmoid function. The ' * ' is denoted as the convolutional operator.

4 EXPERIMENTAL SETUP

This section provides a description of the dataset, the preprocessing method, feature extraction technique, implementation process,

hardware description and various optimizers used for training. Figure 3 shows the overall flowchart of the real-time hand gesture recognition system.

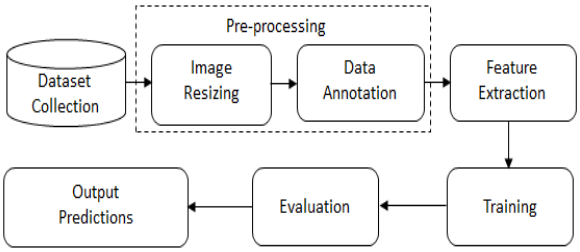

Figure 3: Steps involved in realtime hand gesture recognition system

4.1 Dataset collection

Figure 4: Custom dataset samples captured under unconstrained environments and varying lighting conditions.

Figure 4 shows the dataset collected under unconstrained environments such as different backgrounds, varying light intensity, skin color, hand shape, size, and geometry. The custom dataset is loaded with 7500 samples with ten different classes, where each class has an average of 750 images. The train and test data is split as 80% and 20%, respectively.

4.2 Pre-processing of images

4.2.1. Resizing. The images from the dataset are resized into 300 x 300 pixels. The resize operation is performed by an adaptive interpolation method. The few data points outside the boundaries of the input are filled with a reflection method.

4.2.2. Data Annotation: Annotation is a machine learning process of labeling the data on images containing specific objects. The resized images are annotated to select the proposed region, which consists of an object.

4.3 Feature extraction

To train the CNN for gesture recognition, the features are extracted from the pre-processed images. The features of the proposed region in an image is extracted using the convolutional neural network loaded with the Inception V2 filter banks. The same process is applied for all the images in the database to generate a training pattern.

4.3.1 Inception V2 Architecture. The performance of the neural network is better when the dimensions of the input are not altered drastically by convolutions. The larger convolutions are computationally expensive. Too much reduction in the dimensions of input results in loss of information, known as a "representational bottleneck". The inception V2 model is designed to reduce the dimensionality of its feature map, passing the resulting feature map through a Relu activation function, and then performing the larger convolution [13]. The 1×1 convolutions are incorporated to reduce the dimensionality of the input to large convolutions and made the computations reasonable.

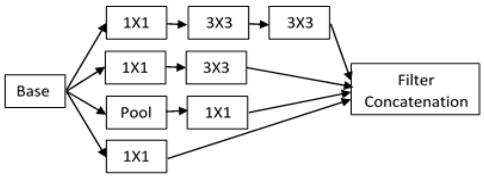

Figure 5: Dimensionality Reduction filter.

To improve the computational speed, 5x5 convolutions from the Inception V1 factorized into two 3x3-convolution operations as shown in Fig 5. This reduces the cost of 5x5 convolution by 2.78 times and leads to a boost in performance. Figure 6 illustrates that a 3x3 convolution made equivalent by performing a 1x3 convolution first, and then performing a 3x1 convolution. This method is 33% inexpensive than the existing 3x3 convolution.

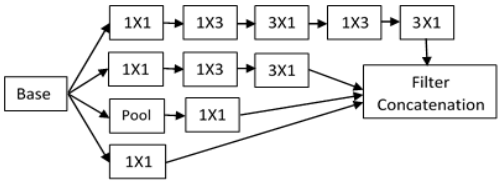

Figure 6: Inception V2 module with deeper filter banks.

The representational bottleneck is minimized by the use of wider filter banks instead of deeper filter banks as illustrated in Fig. 7.

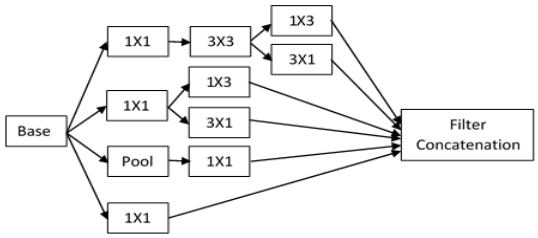

Figure 7: Inception V2 module with wider filter banks.

4.4 Training

The proposed system is trained on a machine with a GPU (NVIDIA GeForce GTX TITAN X (PASCAL)). Tensorflow, CUDA/CuDNN is used for GPU parallelization of the computations. Using the hyperparameters, each training step took about 12 seconds for computing. The model is trained for 35000 steps.

4.4.1 Optimization. The Faster R-CNN Inception V2 model is trained using three different gradient descent optimization techniques such as Momentum optimization, RMSprop optimization, and ADAM optimization algorithm.

4.4.1.1 Momentum Optimization. [14], [15] is used along with stochastic gradient descent (SGD) optimization. Momentum accumulates the gradient from the past steps along with the gradient of the current step, which proposes the direction to move. The summation of a fraction γ of the update vector of the previous time step to the current update vector $v_t = \gamma v_{t-1} + \eta \nabla w_t J(w)$, and $w = w - v_t$.. The momentum term γ is the coefficient of momentum, set as 0.9 or a related value. v_t is the retained gradient. This method results in faster convergence and dampens oscillation.

4.4.1.2 Root Mean Square Propagation (RMSprop). [14], [16] This optimization algorithm minimizes the oscillations in an inconsistent way than the momentum optimization. The learning rate adjustment is automated in RMSprop. It also chooses a distinct learning rate for every parameter. The exponential average of squares of gradient vt = ρvt-1 + (1-ρ)*gt 2, where gt is the gradient at time t along wj. ∇wt= - (η / $\sqrt{(vt + \epsilon)}$)*gt decides the step size for learning, η is the initial learning rate, wt+1 = wt- ∇wt is a step for updating the weight. The average of w_1 is much larger than w_2, therefore the learning rate for w_1 is less than w_2 and its moves towards the minima.

4.4.1.3. Adaptive Moment Estimation (ADAM). [14], [17] is an efficient method for stochastic gradient descent optimization that utilizes first-order gradients with minimal memory requirement. The method incorporates a discrete adaptive learning rate for various estimated parameters of first and second moments of the gradients. It is a computationally efficient method for an undemanding implementation, with lesser memory requirements. ADAM optimizer keeps an exponentially decaying average of past squared gradients, $v_t = \beta 2 v_{t-1} + (1 - \beta 2) g_t^2$ and past gradients, $m_t = \beta 1 m_{t-1} + (1 - \beta 1) g_t$, m_t is the estimates of the first moment and v_t is the estimates of the second moment. m_t and v_t are biased towards zero during the initial steps for less decay rates. This reduces the overall computation time. $\nabla w_t = -((\eta \ m_t / \sqrt{(vt + \epsilon)}) * g_t)$ decides the step size for learning, $w_{t+1} = w_t - \nabla w_t$ is for updating the weight. $\beta 1$, $\beta 2$ are the hyperparameters.

As gradients become sparsed, ADAM optimizer moderately outperforms RMSprop optimizer towards the end of optimization [17]. The Faster R-CNN Inception V2 architecture is trained with different gradient descent optimizers to produce an accurate and precise hand gesture recognition under dynamic environments and varying lighting conditions.

4.5 Evaluation and Prediction

The trained model is tested using the test samples and its performance is evaluated by the parameters such as precision, recall, and F1 score for various IoU ranges. The accuracy of object detection is determined using IoU. Intersection over Union is computed by area encompassed by both the predicted bounding box and the ground-truth bounding box divided by the area of the union, *IoU* = Area of overlap / Area of union . If there is a perfect overlap between the predicted and the ground-truth bounding boxes, then IoU is unity. In general, as long as IoU= 0.5, the prediction is true. For the larger value of IoU's, the prediction is accurate.

5 RESULTS AND DISCUSSION

In this section, the results obtained using the Faster R-CNN Inception V2 architecture for different Gradient descent optimization algorithms such as Momentum, ADAM, and RMSprop optimizers are discussed. Experiments performed using Python programming, mainly taking advantage of the Tensorflow libraries.

The term AP_{all} is the average precision for the overall detected objects with the small, medium and larger size. AP_{medium} is the average precision for the object size less than 96 x 96 pixels. AP_{large} is the average precision for the object size greater than 96 x 96 pixels. Similarly, AR_{medium} and AR_{large} are the average recall for the object lesser and greater than 96 x 96 pixels, respectively. AR_1, AR_{10}, and AR_{100} are the average recall values obtained for various number detections such as 1, 10 and 100.

Method		Faster R-CNN		
Network		Inception V2		
IoU		0.5:0.95		
Optimizer		Momentum	Adam	RMSprop
Average Precision	AP_{all}	0.781	**0.794**	0.781
	AP_{medium}	0.74	**0.746**	0.743
	AP_{large}	0.806	**0.818**	0.803
Average Recall	AR_1	0.825	**0.832**	0.824
	AR_{10}	0.828	**0.833**	0.827
	AR_{100}	0.829	**0.833**	0.827
	AR_{medium}	0.789	**0.791**	0.788
	AR_{large}	0.846	**0.852**	0.845

Table 1: Faster-RCNN Inception-V2 model for different optimizers when batch size=1 and steps=35,000

The Faster R-CNN Inception V2 model is trained for 35,000 steps with minimal learning rate and a batch size of one. From Table 1, the model with ADAM optimizer trained with a learning rate of 0.0002 resulted in higher precision and recall values. For IoU=0.5:0.95 average precision for all size of hand gesture detections obtained are of 0.794 and average recall for 100 detections per class is estimated as 0.833. The average precision and average recall for large size of detections are determined as 0.818 and 0.852, respectively. By using Momentum optimizer, the

model trained with a learning rate of 0.0002 and the model with RMSprop optimizer trained with the learning rate of 0.004.

The model with Adam optimizer shows comparatively better performance and accurate detections of hand gestures with proper localization. From Table 2, the average precision is obtained for various optimizers with different IoU ranges. For IoU = 0.5 and 0.75, the average precision (AP_{all}) obtained for Adam optimizer is of 0.991 and 0.964, respectively. In case of IoU of 0.75, the RMSprop optimizer also resulted in a similar average precision as that of Adam optimizer. For IoU of 0.5:0.95, the ADAM optimizer has resulted in an average precision of 0.794, which is high compared to the other two optimizers.

IoU	Optimizer	Average Precision (AP_{all})
0.5 : 0.95	Momentum	0.781
	ADAM	**0.794**
	RMSprop	0.781
0.5	Momentum	0.988
	ADAM	**0.991**
	RMSprop	**0.991**
0.75	Momentum	0.958
	ADAM	**0.964**
	RMSprop	**0.964**

Table 2: Average Precision for Faster-RCNN Inception-V2 model for different optimizers and different IoU ranges.

From Table 3, the F1-Score of the proposed model with different optimizers are calculated by the average precision and average recall scores for the IoU=0.5:0.95. F1-score is 0.813 for Adam optimizer with a prediction time of 137ms, 0.804 for Momentum optimizer and 0.803 for RMSprop optimizer with a prediction time of 145ms and 140ms respectively. The prediction time is less and high F1-score is obtained for ADAM optimizer 14].

Optimizer	AP	AR	F1-Score	Prediction Time (ms)
Momentum	0.781	0.829	0.804	145
Adam	**0.794**	**0.833**	**0.813**	**137**
RMSprop	0.781	0.827	0.803	140

Table 3: F1-Score of various optimizers from the average precision and average recall values at IoU = 0.5:0.95

The total loss function of the hand gesture recognition model for different optimizers trained for 35,000 steps is shown in Fig 8. It is observed that there is a divulgence of the graphs around 20,000 step. The model is finally trained up to 78,000 steps and there is no divulgence in the loss curve. The loss of the model is reducing for all the optimizers and is maintained below 0.10. When the loss reduces the accuracy of the detection increases.

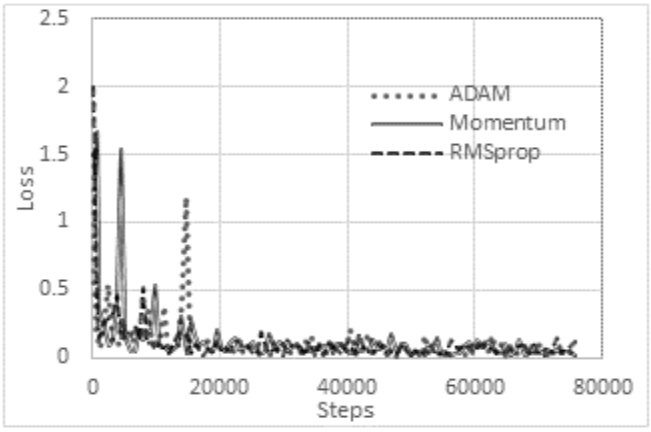

Figure 8: Total loss curve for different optimizers

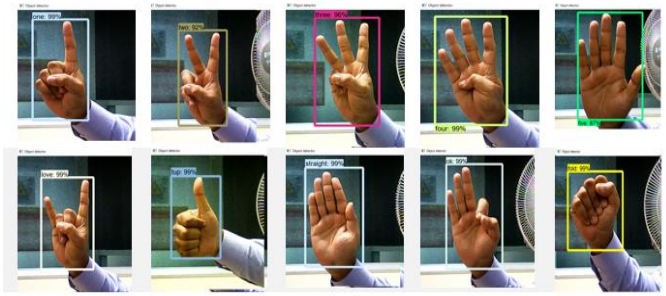

Figure 9: Realtime identification of hand gestures

The realtime output of the Faster R-CNN Inception V2 hand gesture recognition for 10 different gesture is shown in Fig. 9.

6 CONCLUSION

The custom dataset is collected under unconstrained environments such as different lighting conditions and background. This dataset is trained and analyzed using Faster R-CNN Inception V2 model for different gradient descent optimization algorithm for 35,000 steps with learning rate as 0.0002 for both Adam and Momentum optimizers and for RMSprop as 0.004. It is observed that Adam optimizer perform better compared to Momentum and RMSprop optimizer. For Adam optimization algorithm, the average precision, average recall, and F1-score observed for IoU of 0.5:0.95 are 0.794, 0.833 and 0.813, respectively. For true prediction, the average precision obtained as 0.991 with prediction time of 137ms for ADAM optimizer and RMS prop follows closely to ADAM.

ACKNOWLEDGMENT

The authors would like to thank NVIDIA for providing NVIDIA Titan X GPU under the University Research Programme.

REFERENCES

[1] J. Sanchez-Riera, K. Srinivasan, K.-L. Hua, W.-H. Cheng, M. A. Hossain, M. F. Alhamid, "Robust RGB-D hand tracking using deep learning priors", *IEEE Transactions on Circuits and Systems for Video Technology*., Volume: 28 , Issue No : 9 , pages: 2289 – 2301, 2018.

[2] I. Oikonomidis, M.I.A. Lourakis, A.A. Argyros, "Evolutionary Quasi-random Search for Hand Articulations Tracking" *IEEE Conference on Computer Vision and Pattern Recognition*., 2014.

[3] C.Nuzzi, S.Pasinetti, M.Lancini, F.Docchio, G.Sansoni. "Deep Learning based Machine Vision: first steps towards a hand gesture recognition set up for Collaborative Robots", *Workshop on Metrology for Industry 4.0 and IoT*, 2018.

[4] R. Girshick, "Fast R-CNN." *In Proceedings of the IEEE International Conference on Computer Vision*, pages 1440–1448, 2015.

[5] R. Girshick, Donahue, J., Darrell, T., Malik, J.: "Rich feature hierarchies for accurate object detection and semantic segmentation". *In: CVPR*, https://arxiv.org/pdf/1311.2524, 2014.

[6] J. Huang, I. Fischer, Z Wojna, "Speed/accuracy trade-offs for modern convolutional object detectors", *arXiv preprint arXiv:1611.10012*, https://arxiv.org/abs/1611.10012, 2017.

[7] D. Wu, L. Pigou, P.J. Kindermanz et al., "Deep dynamic neural networks for multimodal gesture segmentation and recognition", *IEEE Transactions on Pattern Analysis And Machine Intelligence*, vol. 38, no. 8, pp. 1583-1597, 2016.

[8] Javier O. Pinzon Arenas, Robinson Jimenez Moreno, Paula C. Useche Murillo, "Hand gesture recognition by means of region-based convolutional neural networks" *Contemporary Engineering Sciences*, Vol. 10, 2017, No. 27, 1329-1342.

[9] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards real-time object detection with region proposal networks". *In Advances in neural information processing systems*, pages 91–99, 2015.

[10] H. I. Lin, M. H. Hsu, W. K. Chen, "Human hand gesture recognition using a convolution neural network", 2014 *IEEE International Conference on Automation Science and Engineering (CASE)*, pp. 1038-1043, August 2014.

[11] I. Alnujaim, H. Alali, F. Khan, Y. Kim, "Hand gesture recognition using input impedance variation of two antennas with transfer learning", *IEEE Sensors Journal*, vol. 18, no. 10, pp. 4129-4135, May 2018.

[12] A. Chaudhary, J.L. Raheja, "Light Invariant Real-Time Robust Hand Gesture Recognition", *Optik - International Journal for Light and Electron Optics*, Pages 283–294, (Elsevier), April 2018.

[13] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. "Rethinking the Inception architecture for computer vision". *arXiv preprint, 1512.00567*, https://arxiv.org/pdf/1512.00567, 2015.

[14] Sebastian Ruder, "An overview of gradient descent optimization algorithms", *arXiv preprint arXiv:1609.04747*, https://arxiv.org/abs/1609.04747, 2017

[15] Qian. N, "On the momentum term in gradient descent learning algorithms", *Neural Networks*, Volume 12, Issue 1, January Pages 145-151, 1999,

[16] Tieleman, T., & Geoffrey, H. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude", *Coursera: Neural Networks for Machine Learning* 4.2, 2012.

[17] Kingma, D., & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, https://arxiv.org/abs/1412.6980, 2014.