Effectiveness of Deep Learning in Real Time Object Detection

Faysal Hossain
Institute of ICT
Bangladesh University of Engineering and Technology
Dhaka-1205, Bangladesh

Md. Raihan-Al-Masud
Institute of ICT
Bangladesh University of Engineering and Technology
Dhaka-1205, Bangladesh

M. Rubaiyat Hossain Mondal
Institute of ICT
Bangladesh University of Engineering and Technology
Dhaka-1205, Bangladesh

ABSTRACT
Deep learning based object detection has recently gained significant interest. This work focuses on real time object detection using two deep learning models named Faster Regional Convolution Neural Network (Faster-RCNN) and MobileNet Single Shot MultiBox Detector (MobileNet-SSD). An experiment is done using Python for programming, TensorFlow library for computing and OpenCV for computer vision. The Faster-RCNN and MobileNet-SSD models are trained using 400 images of four objects which are persons, watches, cell phones, and books. It is shown that for the images considered, Faster-RCNN can successfully detect these four objects with higher accuracy than MobileNet-SSD. Faster-RCNN also requires less time than MobilneNet-SSD for training the objects. However, Faster-RCNN model is slightly slower than MobileNet-SSD in real time object detection.

Keywords
Image; object detection; deep learning; Fast-RCNN; CNN.

1. INTRODUCTION
The ability to visually detect and track multiple objects across a scene has been a long-standing challenge within the computer vision and machine learning communities [1-9]. Object detection is useful in the field of automation for example, for self-driving cars, monitoring pedestrians, and traffic signals on the road and experiments in medical science. Some of the common classical methods in object detection are Hough transform, frame-difference, background subtraction, optical flow, sliding window, and deformable part model methods. Compared to the classical methods, the deep learning based object detection methods have strong capability in feature expression and feature learning. Deep learning based approaches [10-17] work in the mode of region selection, feature extraction and classification. Deep learning along with machine learning algorithms [18, 19, 20] and image processing techniques [21-22] have the potential for the prediction of novel coronavirus disease 2019 known as COVID-19 [23, 24, 25]. One algorithm of deep learning is the convolutional neural network (CNN) which is a form of artificial neural network (ANN) [10-12]. In CNN, convolution operation is done to extract features from the input image, pooling is performed to reduce the dimensionality of each feature map, and flattening is done to transform two-dimensional data into one-dimensional.

The work in [2] presents the design details of a face recognition system. CNN algorithm has been used in [7] because of their robustness to real-life scenarios and scalability to training data size. However, accuracy becomes challenging when real time objects of videos are to be recognized. Faster regional CNN (Faster-RCNN) and mobilenet single shot multibox detector (MobileNet-SSD) models are popular object detector models in deep learning field. Faster-RCNN uses the region proposal technique where MobileNet-SSD uses the single forward path multi box technique. Another technique is you only look once (YOLO) [17] which reframes object detection as a single regression issue, starting from pixels to the coordinates of bounding box and classification. While YOLO identifies objects very quickly it suffers from lack of accuracy in the case of some small objects.

This paper describes the algorithms of Faster-RCNN and MobileNet-SSD models for detecting real time objects within images captured by a camera. The performance of these two models are compared using four set of images. The speciality of this research is that instead of using publicly available datasets, this paper creates and then uses a dataset by collecting images of 4 objects: person, watch, cell phone, and book from the Internet. The rest of the paper is organized as follows. Section 2 describes Faster-RCNN and MobileNet-SSD models. The system implementation is presented in Section 3 and performance evaluation is shown in Section 4. The concluding remarks are presented in Section 5.

2. DESCRIPTION OF FASTER-RCNN AND MOBILENET-SSD
This section describes Faster-RCNN and MobileNet-SSD. Firstly, Faster-RCNN is considered. Faster-RCNN is derived from its predecessors regions with CNN (RCNN) [16] and Fast-RCNN [13]. RCNN uses selective search algorithm [14] to extract 2000 region from each image, these regions are then warped and fed into a CNN block to produce a feature vector. The CNN block extracts features and then send to a support vector machine classifier to find the presence of objects. RCNN is time consuming as it has to classify 2000 region proposals per image. For the case of Fast-RCNN, the input image is directly fed to the CNN block which produces a convolutional feature map. This feature map is used to identify region proposals (using selective search) which are then warped into squares. These are then fed to a region of interest (RoI) pooling layer which makes the squares into fixed size output suitable for passing to the next fully connected layers. The output from RoI poling is classified by softmax layer. Fast-RCNN is still slow because of the use of fixed selective search algorithm. Faster-RCNN combats the somewhat complex training pipeline that both RCNN [16] and Fast-RCNN [13] exhibit. Faster-RCNN, is composed of two networks: firstly, region proposal network (RPN) for

generating region proposals and secondly, a network using these proposals to detect objects. The stages of Faster-RCNN are shown in Fig. 1. The entire system is a single, unified network for object detection. The operation of Faster-RCNN is similar to that of Fast-RCNN except that Faster-RCNN uses a separate network to predict region proposals, while Fast-RCNN uses selective search [14] to generate region proposals. The time cost of generating region proposals is much smaller in RPN than selective search, where RPN shares the most computation with the object detection network. Briefly, RPN ranks region/anchor boxes and proposes the ones most likely containing objects [26].

MobileNet-SSD is developed to predict all at once with the bounding boxes and the class probabilities with end-to-end CNN architecture [15]. This model takes an image as input which passes through multiple convolutional layers with different sizes of filters. Feature maps from convolutional layers at different position of the network are used to predict the bounding boxes. They are processed by a specific convolutional layer with 3x3 filters called extra feature layers to produce a set of bounding boxes like to the anchor boxes of the Fast-RCNN [13]. The stages of MobileNet-SSD are illustrated in Fig. 2.

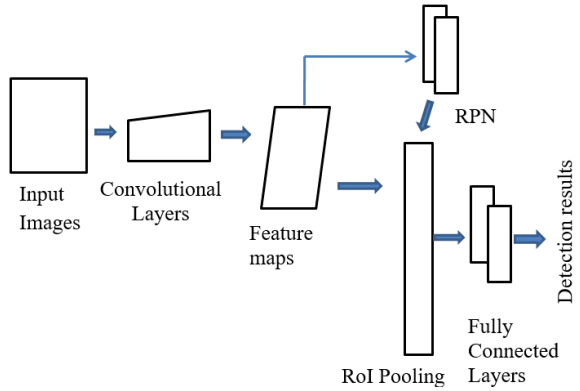


Fig. 1: Typical steps of Faster-RCNN

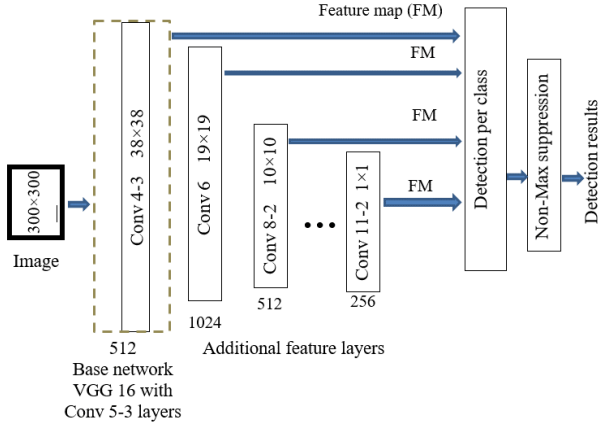


Fig. 2: Typical steps of MobileNet-SSD

3. IMPLEMENTATION

This project was implemented using several software libraries, packages and programs to apply machine learning. Python with TensorFlow was used for the deep learning computations. The GPU version of TensorFlow was used which required extra programs from the GPU designer NVIDIA, such as CUDA 9.0 Toolkit, cuDNN 7.0.5 and their GPU drivers. The card used for this project was a NVIDIA GeForce mx150. OpenCV was applied for computer vision purposes. In addition, Anaconda consisting of Jupiter Notebook and Spyder was used.

In order to create custom object recognized classifier, images of 4 objects: person, watch, cell phone, and book were collected from the Internet as data set. A total of 400 images of four objects were collected which were then divided into two parts as training and test data set. For each of the objects, 70 percent was used for training and 30 percent for testing. First of all, the images were labelled using LabelImage [27] function to classify by the object detection. LabelImgage saved a .xml file containing the label data for each image. The .xml files were used to generate TFRecords, which were one of the inputs to the TensorFlow trainer. For this purpose, the xml_to_csv.py and generate_tfrecord.py scripts were used to create .csv files containing all the data for the train and test images. Next a label map was created and the training configuration file was edited. The configurations were then performed for the object detection training pipeline which defined the model and parameters for training.

The progress of the training job was viewed by the use of TensorBoard which provides information and graphs about the progression of the training. The loss graph showed the overall loss of the classifier over time. When the training was completed, the last step was to generate the frozen inference graph (.pb file). The .pb file contained the object detection classifier. With the use of the library called CV2, a video feed was opened with the window size of 800 by 600 pixels. It drew a box around the object that was found, wrote what type of object it was and the confidence of the identification being correct.

Each step of training reported the loss which started high and got lower as the training progressed. For the training of Faster-RCNN, the Faster-RCNN-Inception-V2 model was used. The total loss of Faster-RCNN model was reduced to below 0.1 by completing almost 17 thousand steps within 3 hours training. For MobileNet-SSD, the loss was reduced to below 1.5 after completing 25 thousand steps in approximate 6 hours and 30 minutes time duration.

Training Algorithm:

Input: Image of four objects: person, book, cell phone, watch

Output: Trained classifier for object detection inference graph

Process:

1. Collect image of four objects: person, book, cell phone and watch
2. Split image data into 70% for training and 30% for testing
3. Label image by LabelImage tools that generate .xml files
4. Convert .xml files into two separate *train_labels.csv and test_labels.csv files*
5. *Convert train_labels.csv to train.record and test_labels.csv to test.record files*
6. Create label map for four objects in labelmap.pbtxt file format
7. Configure the object detection classifier Faster-RCNN/MobileNet-SSD for four objects by changing the number of classes, changing the fine_tune_checkpoint path, changing TensorFlow record path and changing the

label map path

8. Train the object detection classifier Faster-RCNN/ MobileNet-SSD
9. Observe the training accuracy and loss on TensorBoard
10. Train the model until good accuracy is obtained
11. Stop the training when good accuracy is achieved
12. Create frozen inference graph from trained classifier

Real time Evaluation Algorithm:
Input: Real time pixel of objects

Output: Recognition the objects

Process:

1. Import necessaries libraries such as numpy, OpenCV, TensorFlow
2. Create TensorFlow graph with *tf.Graph()* function
3. Load the object detection frozen inference graph into memory with TensorFlow graph
4. Load the object detection label map
5. Configure the camera through OpenCV by *cv2.VideoCapture()* function
6. Create TensorFlow Session with object detection frozen inference graph
7. Capture real time pixels through OpenCV and camera by *cap.read()* function
8. Expand the captured pixels into a shape [frame, weight, height, channel]
9. Initialize five tensors: image_tensor, boxes, scores, classes, num_detections
10. Feed pixel data into the detection graph and run the TensorFlow session.
11. Visualize the output of the object detection model through OpenCV
12. Press 'q' to destroy window and to release camera

4. PERFORMANCE EVALUATION

This section compares the performance of Faster-RCNN and MobileNet-SSD in detecting objects from real time captured images. During the training period of these two models, the loss is reduced below a threshold. The loss versus iteration of Faster-RCNN is shown in Fig. 3. It can be seen that the loss is blow 0.1 at around 17k steps. The loss versus iteration of MobileNet-SSD is shown in Fig. 4. It can be seen that the loss is below 1.5 approximately at 25k steps. Therefore, compared to MobileNet-SSD, Faster-RCNN requires less time and less steps to obtain low loss that is more accuracy during training of objects.

When training is completed, both the models were tested through camera of a laptop. The testing device had the same configuration as the training device. The output of the Faster-RCNN is shown in Fig. 5. There are six images marked as (a), (b), (c), (d), (e) and (f) in Fig. 5. In the images of Fig. 5, (a) contains a person and a cell phone, (b) contains a person and a watch, (c) contains a person, a cell phone, and a watch, (d) contains a person and a book, (e) contains a person, a book and a person within a book, and (f) contains a person, book, watch, and a person within a book.

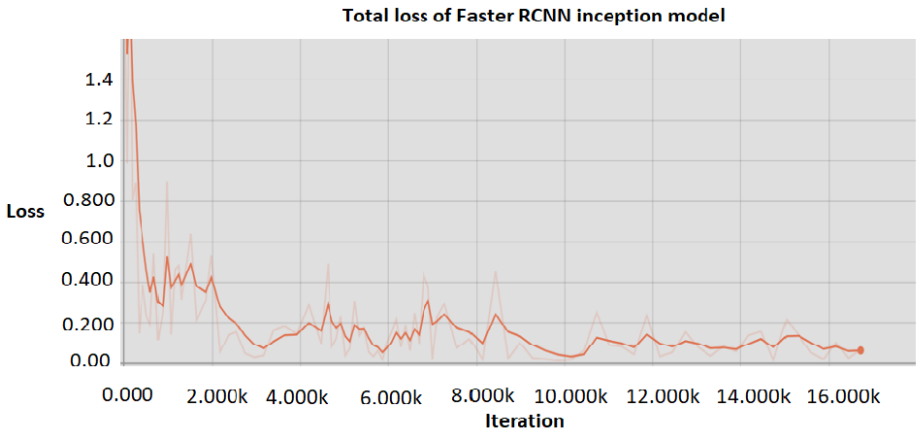


Fig 3: Total loss of Faster-RCNN.

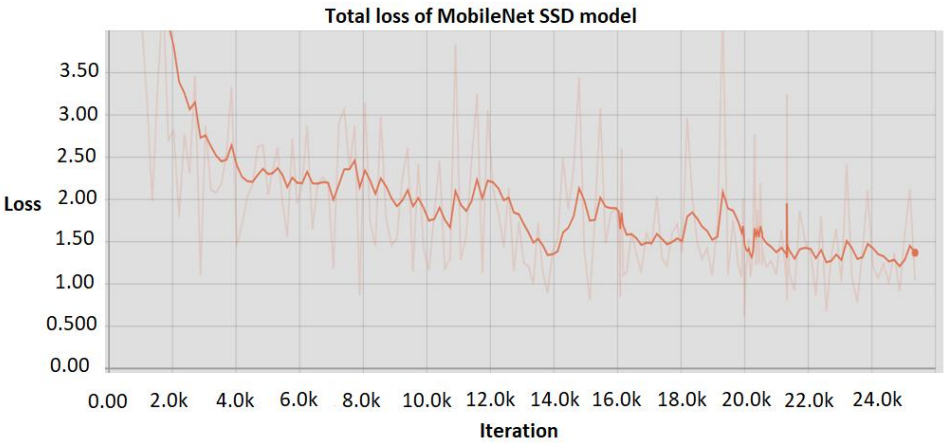
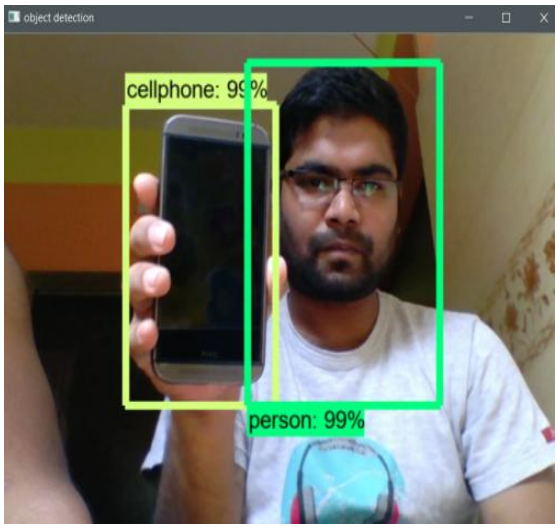
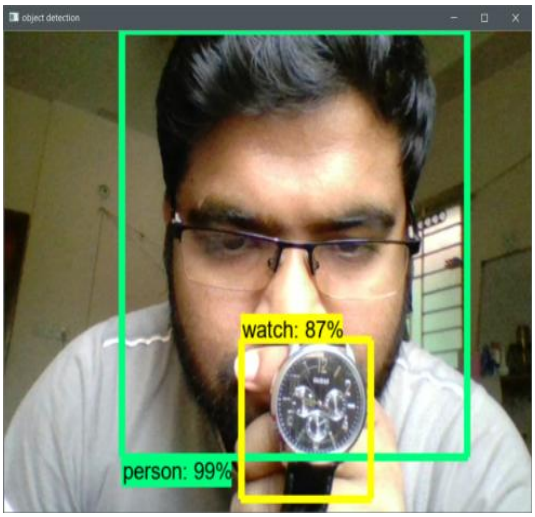


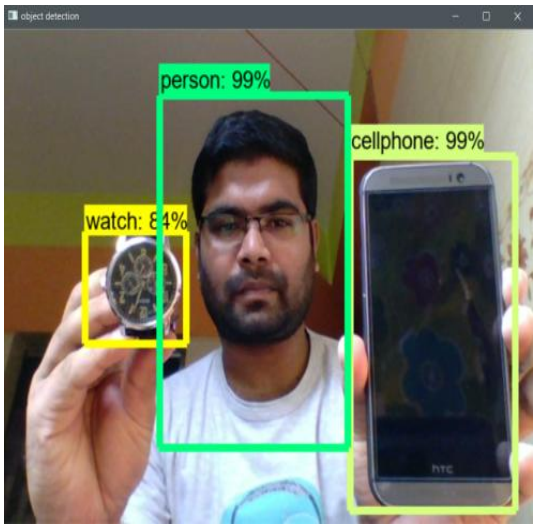
Fig 4: Total loss of MobileNet-SSD model



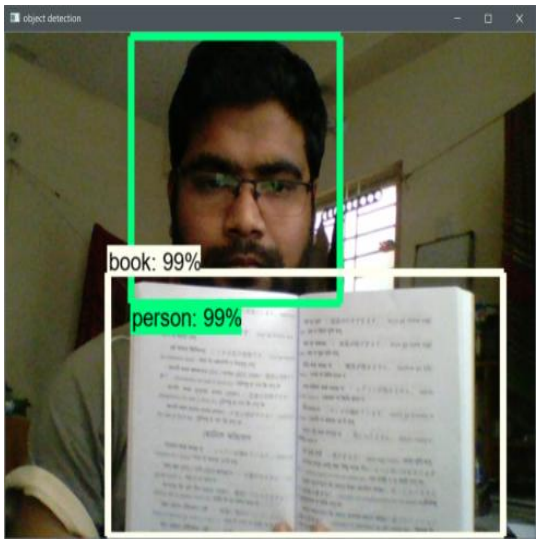
(a)



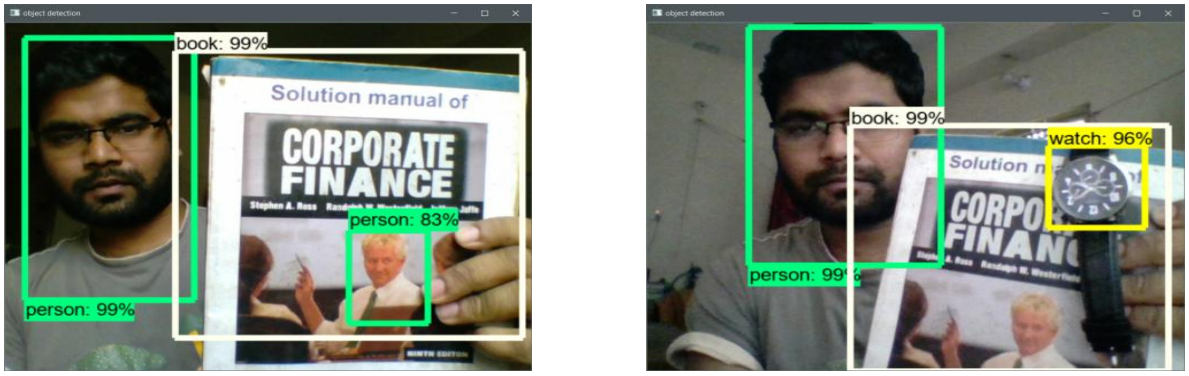
(b)



(c)



(d)



(e) (f)

Fig 5: Illustration of real time image detection using Faster-RCNN method

It can be seen from Fig. 5 that Faster-RCNN model successfully detects the objects with a high score. For instance, a person and a cell phone are detected with an accuracy of 99% by Faster-RCNN as shown in the image of Fig. 5(a). Similarly, a person, a watch and a cell phone are detected with accuracies of 99%, 84% and 99%, respectively, by Faster-RCNN as shown in the image of Fig. 5(c). The same six images in Fig. 5 are used to detect the object using MobileNet-SSD. Therefore, separate figures with score are not illustrated for MobileNet-SSD. From our experiments it is found that MobileNet-SSD cannot detect the objects accurately in some cases. For instance, for the image in Fig. 5(c), MobileNet-SSD cannot detect a watch, while it detects a person at 59% accuracy and a cell phone at 98% accuracy. For the case of Fig. 5(f), both Faster-RCNN and MobileNet-SSD fail to detect the person within a book. However, Faster-RCNN detects the other objects in that image with higher accuracy than MobileNet-SSD. So, the accuracy of the Faster-RCNN model is higher than MobileNet-SSD model in this scenario.

Table 1 summarizes the accuracy of these two models in detecting objects of the six images illustrated in Fig. 5. It can be seen that the confidence score or the accuracy in detecting the objects is greater for Faster-RCNN in all the six cases. Hence, Faster-RCNN model is more reliable than MobileNet-SSD model. It is already shown that in the training period, Faster-RCNN takes significantly less time and less steps to achieve more accuracy than MobileNet-SSD. However, Faster-RCNN model is slightly slower than MobileNet-SSD model in real time response.

Table 1: Reliability of the models

Model	Confidence score of objects
Faster-RCNN	(a) Person: 99%, cell phone: 99%
	(b) Person: 99%, watch: 87%
	(c) Person: 99%, watch: 84%, cell phone: 99%
	(d) Person: 99%, book: 99%
	(e) Person: 99%, book: 99%, person in a book: 83%
	(f) Person: 99%, book: 99%, watch: 96%, person in a book not detected
MobileNet-SSD	(a) Person: 76%, cell phone: 86%
	(b) Person: 96%, watch wrongly detected as a cell phone
	(c) Person: 59%, watch not detected, cell phone: 98%,
	(d) Person: 89%, book: 90%
	(e) Person 96%, book wrongly detected as a cell phone, person in a book not detected
	(f) Person: 99%, book: 56%, watch: 90%, person in a book not detected

5. CONCLUSIONS

In this paper, the effectiveness of Faster-RCNN and MobileNet-SSD methods are studied for real time object detection. Firstly, the algorithms of these two models are described. Performance evaluation is done using a training dataset of 400 images of four different objects which are persons, watches, cell phones, and books. During training, Faster-RCNN takes significantly less time and less steps to achieve more accuracy than MobileNet-SSD. However, during real time detection, Faster-RCNN model has better accuracy at the cost of lower response compared to MobileNet-SSD. Moreover, in a number of cases, MobileNet-SSD fails to detect objects in real time, while Faster-RCNN can detect successfully. For instance, a person, a watch and a cell phone within a real time captured image are detected by Faster-RCNN with accuracies of 99%, 84% and 99%, respectively. On the other hand, for the same captured image, MobileNet-SSD cannot detect the watch, while it detects the person at 59% accuracy and the cell phone at 98% accuracy. In future, the performance of these models will be improved by using more training data, using higher resolution cameras and using powerful Nvidia GPU.

6. ACKNOWLEDGMENTS

A part of this research is from the postgraduate diploma (ICT) project of the first author (Faysal Hossain) conducted under the supervision of the third author (M. Rubaiyat Hossain Mondal) at the Institute of Information and Communication Technology (IICT) of Bangladesh University of Engineering

and Technology, Bangladesh. Therefore, the authors would like to thank IICT, BUET for its support.

7. REFERENCES

[1] Mathe, S., and Sminchisescu, C., "Actions in the eye: dynamic gaze datasets and learnt saliency models for visual recognition," *in IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1408-1424, July 1 2015. doi: 10.1109/TPAMI.2014.2366154.

[2] Chellappa, R. et al., "Towards the design of an end-to-end automated system for image and video-based recognition," *Information Theory and Applications Workshop*, La Jolla, CA, 2016, pp. 1-7.

[3] Nikan, S. and Ahmadi, M., "Effectiveness of various classification techniques on human face recognition," 2014 International Conference on High Performance Computing & Simulation (HPCS), Bologna, 2014, pp. 651-655. doi: 10.1109/HPCSim.2014.6903749.

[4] Wright, J., Yang, A. Y., Ganesh, A., Sastry, S. S. and Ma, Y., "Robust face recognition via sparse representation," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, pp. 210-227, February 2009.

[5] Huang, G. B., Zhou, H., Ding, X. and Zhang, R., "Extreme learning machine for regression and multiclass classification," IEEE Trans. Syst., Man, Cybern., Syst., vol. 45, pp. 513-529, April 2012.

[6] Nikan, S. and Ahmadi, M., "Study of the Effectiveness of Various Feature Extractors for Human Face Recognition for Low Resolution Images," in: Proc. International Conf. on Artificial Intell. and Software Eng. (AISE14). Phuket, pp. 1-6, January 2014.

[7] Wu, J., Ma, L. and Hu, X., "Delving deeper into convolutional neural networks for camera relocalization," 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 2017, pp. 5644-5651. doi: 10.1109/ICRA.2017.7989663.

[8] N. Kumar and A. Sethi, "Fast Learning-Based Single Image Super-Resolution," in *IEEE Transactions on Multimedia*, vol. 18, no. 8, pp. 1504-1515, Aug. 2016. doi: 10.1109/TMM.2016.2571625.

[9] C. M. Bishop, "Pattern Recognition and Machine Learning (Information Science and Statistics)". *Springer-Verlag New York, Inc., Secaucus, NJ*, USA, 2006.

[10] S. Ren, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks", *CoRRabs/1506.01497(2015).* URL: *http://arxiv.org/abs/1506.01497*.

[11] N. Ketkar, "Deep Learning with Python: A Hands-on Introduction", *Bangalore, Karnataka, India, ISBN-13 (electronic)*: 978-1-4842-2766-4, DOI 10.1007/978-1-4842-2766-4.

[12] I. Goodellow, Y. Bengio, A. Courville, "Deep Learning (Adaptive Computation and Machine Learning)", *An MIT Press Book*, URL: https://www.deeplearningbook.org/ Accessed: 2018-05-28.

[13] G. Ross, "Fast R-CNN", *Proceedings of the IEEE International Conference on Computer Vision.* 2015, pp. 1440–1448.

[14] J. R. R. Uijlings, et al, "Selective search for object recognition", URL: http://disi.unitn.it/ uijlings/SelectiveSearch.html.

[15] W. Liu, C. Szegedy, SSD: Single Shot MultiBox Detector, In *arXiv:1512.02325.*

[16] R. Girshick, J. Donahue, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation Tech report (v5), URL: http://*arxiv.org: 1311.2524.*

[17] J. Redmon, S. Divvala, Girshick, R., and Farhadi, A., "You only look once: unified, real-time object detection," IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, USA, 2016, pp. 779-788.

[18] Bharati S., Podder P., and Mondal, M. R. H., "Artificial Neural Network Based Breast Cancer Screening: A Comprehensive Review", International Journal of Computer Information Systems and Industrial Management Applications, MIR Labs, USA, vol. 12 (2020), pp. 125-137, May 2020.

[19] Bharati S., Podder P., and Mondal, M. R. H., Diagnosis of Polycystic Ovary Syndrome Using Machine Learning Algorithms. Presented at 2020 IEEE Region 10 Symposium (TENSYMP), 5-7 June 2020, Bangladesh.

[20] Masud, M. R. A., and Mondal, M. R. H., "Data-Driven Diagnosis of Spinal Abnormalities Using Feature Selection and Machine Learning Algorithms," in PLOS One, 15(2): e0228422, Feb 2020,https://doi.org/10.1371/journal.pone.0228422.

[21] Kabir, M. A., and Mondal, M. R. H., "Edge-Based and Prediction-Based Transformations for Lossless Image Compression", Journal of Imaging, vol. 4, no. 5, DOI: 10.3390/jimaging4050064, May 2018.

[22] Kabir, M. A., and Mondal, M. R. H., "Edge-based Transformation and Entropy Coding for Lossless Image Compression". International Conference on Electrical, Computer and Communication Engineering (ECCE 2017), Cox's Bazar, Bangladesh, Feb 2017.

[23] Khanam F., Nowrin I., and Mondal M. R. H., "Data Visualization and Analyzation of COVID-19", Journal of Scientific Research and Reports, vol. 26, no. 3, pp. 42-52, Apr. 2020, https://doi.org/10.9734/jsrr/2020/v26i330234.

[24] Mondal, M. R. H., Bharati, S., Podder, P., Podder, P., "Data Analytics for Novel Coronavirus Disease", Informatics in Medicine Unlocked, Elsevier, 2020, 100374, https://doi.org/10.1016/j.imu.2020.100374.

[25] Bharati S., Podder P., Mondal M.R.H., Hybrid deep learning for detecting lung diseases from X-ray images, Informatics in Medicine Unlocked, Elsevier, Volume 20, 2020, 100391, ISSN 2352-9148, https://doi.org/10.1016/j.imu.2020.100391.

[26] Stanford Lecture: http://cs231n.github.io/. Accessed: 2018-05-28.

[27] LabelImage, https://github.com/tzutalin/labelImg. Accessed:2018-05-28.