International Journal of Electrical and Computer Engineering (IJECE)
Vol. 9, No. 4, August 2019, pp. 2548~2555
ISSN: 2088-8708, DOI: 10.11591/ijece.v9i4.pp2548-2555

Performance evaluation of fine-tuned faster R-CNN on specific MS COCO objects

Garima Devnani, Ayush Jaiswal, Roshni John, Rajat Chaurasia, Neha Tirpude
Department of Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, India

Article Info

Article history:

Received Jun 4, 2018
Revised Jan 4, 2019
Accepted Mar 4, 2019

Keywords:

Average precision
Convolutional neural network
Deep learning
Fine-tuning
Object detection
Performance evaluation

ABSTRACT

Fine-tuning of a model is a method that is most often required to cater the users' explicit requirements. But the question remains whether the model is accurate enough to be used for a certain application. This paper strives to present the metrics used for performance evaluation of a Convolutional Neural Network (CNN) model. The evaluation is based on the training process which provides us with intermediate models after every 1000 iterations. While 1000 iterations is not substantial enough over the range of 490k iterations, the groups are sized with 100k iterations each. Now, the intention was to compare the recorded metrics to evaluate the model in terms of accuracy. The training model used the set of specific categories chosen from the Microsoft Common Objects in Context (MS COCO) dataset, while allowing users to use their own externally available images to test the model's accuracy. Our trained model ensured that all the objects are detected that are present in the image to depict the effect of precision.

Corresponding Author:

Garima Devnani,
Department of Computer Science and Engineering,
Shri Ramdeobaba College of Engineering and Management,
Ramdeo Tekdi, Gittikhadan, Katol Road, Nagpur, Maharashtra, India.
Email: garimadevnani@gmail.com

1. INTRODUCTION

In today's fast paced and technologically savvy world, object detection has created a niche for itself. The applications rendered for the same focus on knowledge acquisition, anomaly detection. Some applications to mention are detection and tracking for unmanned aerial vehicles as presented by Kamate and Yilmazer [1]; and human-robot interaction in Industry 4.0 factories as explained by Sonntag et al. [2]. The architecture of these applications works only with usage of object detection algorithms.

Also known as parameter fine-tuning, it is a transfer learning technique, wherein, a network is pre-trained on a different data set and then retrained on the target set (of categories) as defined by Becherer et al. [3]. Research by Yosinski et al. has shown this method boosts the performance of a network over random initialization [4]. This technique is considered to be advantageous because of its prominent feature of allowing us to limit usage of the number of categories to a specific set that helps tend to the need of a certain application. Fine-tuning gives us a desirable output when trying to overcome problems like:

- When we use R-CNN by Girshick et al. [5], the bounding boxes (BBs) are generated. Theses boxes created, for the objects, will have some part of the background of the image present in it. This background dependence generated can easily mislead categorization and detection as claimed by Sonntag et al. [6].
- The Faster R-CNN by Ren et al. [7] works rapidly when full image convolutional features are shared with the detection network used in [6].

With the algorithms of object detection having undergone multiple levels of evolution over the years, the accuracy of the same has only improved. But the implementations of the CNNs seen over the years do not discuss the accuracy and other performance metrics when we are concerned with only a few of the

categories in object detection. Also, the number of iterations performed for training the model that can act as a factor in the refinement of the accuracy of the model haven't been considered before.

The use of a specific set of categories for training of an object detection algorithm is what is termed as fine-tuning in layman's speech. Girshick et al. [9] explained that the application of the fine-tuning technique on a model for the target dataset shows improvement in the mean average precision. The working of fine-tuning on the VGG_CNN_M_1024 model was further demonstrated in [6], using the subsets of the dataset of the Microsoft Common Objects in Context (MS COCO) dataset as decided by the user as done by Lin et al. [8].

In our approach, we use the path adopted in [6], branching from the work accomplished by Ren et al. [7] which utilizes appraising the joint training wherein the region proposal network is trained jointly with the fast R-CNN by Girshick [9], giving the results with significant improvement in speed. This demonstrates fine-tuning, and is then modified for ensuring that all the categories mentioned in the subset for fine-tuning are detected simultaneously when given a testing images for object detection. The code used is forked and is available in Python language at the following link: https://github.com/ayushjaiswal22/Synapse

2. IMPLEMENTATION

For our purposes, we have used the py-faster-rcnn-ft (https://github.com/DFKI-Interactive-Machine-Learning/py-faster-rcnn-ft) model in Sonntag et al. [6], which is the fork of py-faster-rcnn (https://github.com/rbgirshick/py-faster-rcnn) by Ren et al. [7] allowing us an appropriate application of the fine-tuning technique on the VGG_CNN_M_1024 model for specific object categories of the MS COCO dataset listed by Lin et al. [8].

2.1. Choosing Categories from MS COCO Dataset

Microsoft COCO: Common Object in Context is a large-scale dataset that addresses three core research problems in scene understanding: detecting non-iconic views (or non-canonical perspectives [10]) of objects, contextual reasoning between objects and the precise 2D localization of objects as explained by Lin et al. [8]. The dataset that has been used, was released in 2014, covers over 80 categories of objects. The dataset consists of nearly 164k images, and is divided into the collections-training, validation and testing-segregation by Lin et al. [8]. Table 1 specifies number of images in each collection.

Table 1. Number of images in MS COCO 2014 dataset

Collection	Training	Validation	Testing
Number of images	82,783	40,504	40,775

There exists nearly 886k instances in images of the training + validation collections made by Lin et al. [8]. There liberty to choose any subset of categories from the available dataset. The dataset has annotations available to select categories using category IDs assigned by Lin et al. [8], which can be fetched using the program written in [6]. For our performance evaluation, we have chosen the following categories of pet animals and cattle animals along with the humans: Person, Cat, Dog, Horse, Sheep, and Cow.

2.2. Training the model

2.2.1. Installation and setup

As suggested in [6], for training the model, the installation of the Ubuntu 17.04 64-bit operating system is suitable because this grants an easy and quick installation providing minimal compilation process from supplementary sources. A machine with at least one GPU that supports CUDA6 is required for the functioning of `py-faster-rcnn-ft`. A single Nvidia GTX 1080 was declared sufficient for the execution using the VGG CNN M 1024 model in [6].

For training the model, there was usage of Google Cloud Platform's virtual machine with the Ubuntu 17.04 64-bit operating system and one GPU with the support of CUDA8. A single Nvidia Tesla K80 GPU proved to be sufficient for the training purpose.

2.2.2. Prerequisites

The fine-tuned model to be generated, will be attained using the VGG-16 model. This Faster R-CNN model of Ren et al. is pre-trained on all the 80 categories of the MS COCO dataset. The MS COCO dataset needs to be downloaded separately on the machine to be used for training and executing the fine-tuning of an already pre-trained model.

2.2.3. Intermediate snapshots

When training the model, for n number of iterations, the program creates intermediate snapshots of the model for every 1000 iterations of the algorithm. These snapshots will be tested and used to determine the analysis on how the average precision changes over the 490k iterations run for training our model. We will be using the intermediate models generated after every 100,000 iterations: 100k, 200k, 300k, 400k, and 490k. Figure 1 showcases the use of the Faster R-CNN model designed by Ren et al. [7], to apply the fine-tuning technique in [6] and further evaluate the model for performance analysis.

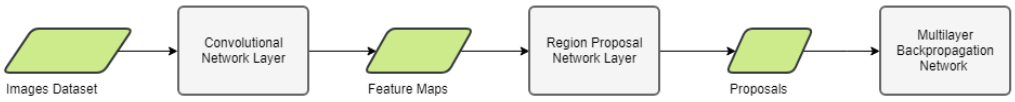

Figure 1. Basic flow of the Faster R-CNN Model

2.3. Methodology

The methodology has been adopted via the use of the VGG-16 Faster R-CNN modelled by Ren et al. [7] and is incorporated with the changes made to apply the fine-tuning technique. Figure 2 depicts the basic architecture applied on the Faster R-CNN model of Ren et al. When training of the model begins, the first step is to fetch the relevant images from the MS COCO training dataset. The term relevant images corresponds the images whose category IDs have been specified in the configuration file. The category IDs chosen are user specific and can vary from person to person.

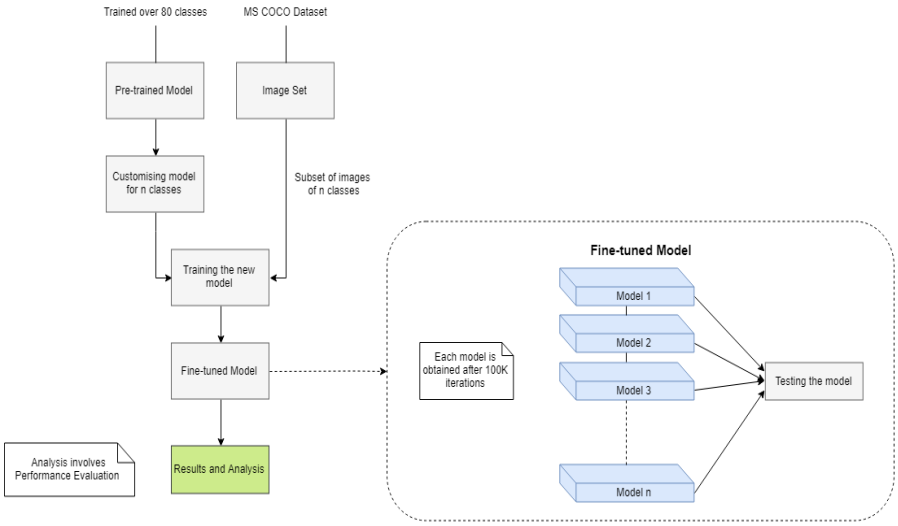

Figure 2. Flow of the Fine-tuning and analysis of the Faster R-CNN model

2.3.1. Convolutional network layer

This specific array of images is then passed on to the convolutional network, wherein the images are processed to obtain feature maps after the last convolutional layer completes its execution as mentionedby Ren et al. [7]. The feature maps attained are only for those categories which have been specified by the user.

2.3.2. Region proposal network

The feature maps acquired are next processed through the sliding window run spatially to generate proposals. For each sliding window of the size 3×3, a group of 9 anchors are generated with the same center (x_a, y_a). These proposals for the anchors are generated with 3 different aspect ratios (1:1, 1:2, and 2:1) and 3 different scales (128, 256, and 512) similar to work of Ren et al. This entire process occurs at the layer consisting of the Region Proposal Network (RPN), generating Regions of Interest (RoI). This was developed to substitute the selective search being executed in the Fast R-CNN of Girshick.

The introduction of aspect ratios and scales of objects came into picture during the construction of the RPN in Faster R-CNN. This was included to handle the different sizes and aspect ratios of the object detected in the images. For each of these anchors, p^* is evaluated to indicate the amount of overlap with the ground-truth bounding boxes,

$$p^* = \begin{Bmatrix} 1 & if\ IoU > 0.7 \\ -1 & if\ IoU < 0.3 \\ 0 & otherwise \end{Bmatrix} \quad (1)$$

where, IoU is intersection over union and is expressed as below:

$$IoU = \frac{Anchor\ \cap\ Ground-truth\ Box}{Anchor\ \cup\ Ground-truth\ Box} \quad (2)$$

Finally the 3×3 spatial features are extracted from the derived feature maps to be forwarded to a smaller network as of Ren et al. [7].

2.3.3. Multilayer backpropagation network

This smaller network accomplishes the following two tasks to obtain the final output, i.e., Classification (cls) and Regression (reg), as created by Ren et al. [7].

The task of regression generates the dimensions for the predicted bounding box for a certain detected object. The dimensions are given as (x, y, w, h), where the each of the mentioned variables are defined as follows:

- x – The x coordinate of the center of the bounding box,
- y – The y coordinate of the center of the bounding box,
- w – The width of the bounding box,
- h – The height of the bounding box.

As Ren et al. [7] have explained, the task of classification derives the probability p indicating whether the box obtained from the regression task contains an object or does it hold the background (0 for no object). This layer consisting of the smaller network is where the loss is calculated over the run of the iterations, backpropagation occurring after every 200 iterations. The loss function for every anchor i is expressed as follows:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} L_{reg}(t_i, t_i^*) \quad (3)$$

In equation (3), the 2 terms in addition are normalized by N_{cls} and N_{reg} and weighted by a balancing parameter λ. For our implementation as taken by Ren et al. [7], the cls term is normalized by the mini-batch size of the value $N_{cls} = 256$ and the reg term is normalized by the number of anchor locations nearly $N_{reg} \sim 2{,}400$. The default value of $\lambda = 10$, which sets the terms cls and reg as roughly equally weighted, as decided by Ren et al. [7]. The values t and t^* are calculated in the form (x, y, w, h), as follows:

$$t = \left[\frac{x - x_a}{w_a}, \frac{y - y_a}{w_a}, \log\frac{w}{w_a}, \log\frac{h}{h_a} \right] \quad (4)$$

$$t^* = \left[\frac{x^* - x_a}{w_a}, \frac{y^* - y_a}{w_a}, \log\frac{w^*}{w_a}, \log\frac{h^*}{h_a} \right] \quad (5)$$

where, anchor's properties, defined by Ren et al. [7], are depicted by:

- x_a – The x coordinate of the center of the anchor's bounding box,
- y_a – The y coordinate of the center of the anchor's bounding box,
- w_a – The width of the anchor's bounding box,
- h_a – The height of the anchor's bounding box,

and the properties of ground-truth bounding box [5] are expressed with the following:
- x^* - The x coordinate of the center of the bounding box,
- y^* - The y coordinate of the center of the bounding box,
- w^* - The width of the bounding box,
- h^* - The height of the bounding box.

2.4. Image selection for testing

For the testing, *tools/demo2.py* is the program that has been executed. In [6], a version of the procedure *tools/demo.py* was used for the visualizing of the fine-tuned trained model. In their version, the user is not required to manually search for applicable images in the dataset to be used for the testing on the model in [6].

In our version, the images the user wished to be tested can be input externally, i.e., the images need not necessarily be taken from the dataset but can be any image they own or have downloaded from other external sources. Running the *tools/demo2.py*, when the image is given for testing, the model detects all the objects present in the image simultaneously, which means it accomplishes a comprehensive object detection for our specific set of categories the model has been trained on.

3. PERFORMANCE EVALUATION

In the MS COCO dataset, there are nearly 164k images and each of these images contain instances of multiple classes. Also, the model uses convolution to detect features which makes the model encounter thousands of negative i.e., background examples for every positive example.

In object detection, the process is based on the location of the object in the image. Every bounding box detected is assigned to the ground truth objects to determine whether it is a true or false positive, depending on the amount of overlapping. For detection, many bounding boxes are created by the model with a probability or confidence score attached to it. If there are multiple bounding boxes for the same image, the bounding box with the highest score is greedily selected [11].

The detection is considered to be correct when the area of overlap between detected bounding box B_p and the ground truth bounding box B_{gt} exceed 50%, as claimed by Girshick [9], also called as Intersection of Union (IoU). The formula is as follows:

$$A = \frac{area(B_p \cap B_{gt})}{area(B_p \cup B_{gt})} \tag{6}$$

3.1. Precision

Percentage of true positives (tp) in the resultant bounding box, where fp is false positives and n is the total number of bounding boxes retrieved.

$$Precision = \frac{tp}{tp + fp} = \frac{tp}{n} \tag{7}$$

3.2. Recall

Percentage of the actual objects detected where fn is false negatives.

$$Recall = \frac{tp}{tp + fn} \tag{8}$$

4. RESULTS AND FINDING

4.1. Mean average precision

Table 2 shows the recorded mean average precision (mAP) of the six categories chosen from the MS COCO dataset for the intermediate snapshots that were evaluated. The values depict that over the 490k iterations, the model steadily improves itself for the specific set of categories.

The numbers in Table 2 represent that even the intermediate snapshots or models created can be used for testing, since they have a raise in accuracy compared to the last intermediate snapshot. The numbers helps us compare the final results w.r.t. the entire model.

Table 2. Mean average precision (mAP) @ IoU = [0.50, 0.95] over the interval of 100k iterations

Object	mAP (Mean Average Precision)				
	100k Iterations	200k Iterations	300k Iterations	400k Iterations	490k Iterations
Entire Model	15.5	17.5	19.3	25.3	26.2
Person	22.0	22.7	24.8	28.9	29.3
Cat	21.5	28.9	29.0	37.3	38.9
Dog	13.5	13.0	16.7	24.8	24.5
Horse	16.5	17.0	18.7	25.3	27.0
Sheep	11.3	11.4	14.3	19.9	20.4
Cow	7.9	12.0	12.2	15.9	17.1

4.2. Loss function

Using (3), we calculate the loss after every 100k iterations. The results show that the loss value is very low at the end of the training, while experiencing some fluctuations during the iterations, as depicted in Figure 3.

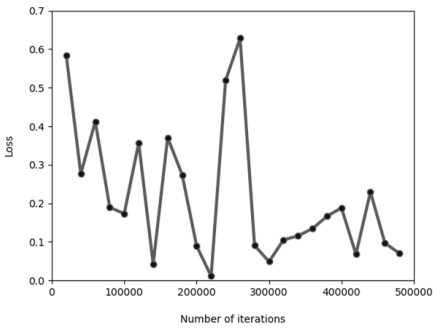

Figure 3. Loss values vs. number of iterations on the MS COCO 2014 train set

4.3. Precision-recall curve

The Precision-Recall Curve, plotted using the calculations documented with (7) and (8) in Figure 4, shows a steady growth over the span the training of the model.

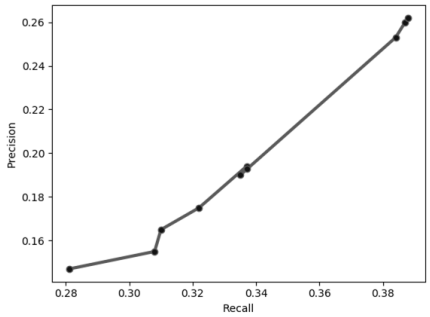

Figure 4. The Precision-Recall Curve @ IoU = [0.50, 0.95] for all sized objects over 490k iterations

4.4. Precision of IoU prediction

Precision of IoU is a measure to find the tightly bound box for the objects detected using feature maps. While the "all" attribute defines the box being placed over all sized objects in the image, the objects sizes detected can be of the following types:

- Small,
- Medium, and
- Large.

Figure 5 depicts the variations in precision when considering the sizes of the objects. As expected the object sized as large have the highest precision, while the small sized are difficult to detect. But, when all of the sizes are considered together, the average precision is at par with the average precision of the model.

The other parameter used w.r.t. measuring precision of IoU is the area percentages when all sizes of objects are considered in the area and they are as follows: Range of 50 – 95%, 50%, and 75%. Figure 6 shows that when IoU area is at least 50% the precision is measured to be the highest.

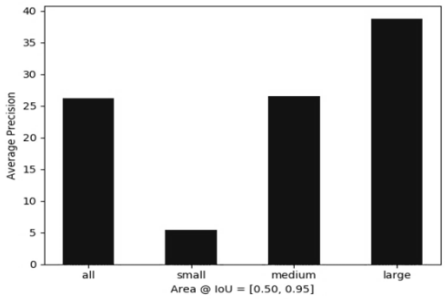

Figure 5. Average Precision vs Area @ IoU = [0.50, 0.95] for objects sized as [all, small, medium, large]

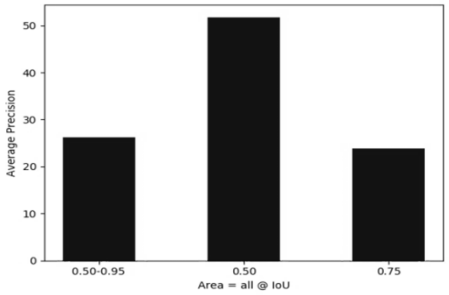

Figure 6. Average Precision vs Area @ IoU for all sized objects

5. CONCLUSION

From the above derived results, we can conclude that the Faster R-CNN (VGG model) maintains its accuracy when trained with the fine-tuning technique as well. The average precision, for each class the model is trained on, depicts a steady escalation, ensuring correct detection of object by the model.

Furthermore, over the course of training the model, the loss values depict fluctuations but by the end of the process, it has the lowest possible loss incurred, representing the model's better working and accuracy. The growth of the Precision-Recall curve shown in Figure 3 means that the model detects the actual objects correctly, inside the bounding box. This implies that the accuracy w.r.t average precision and recall, inside the bounding box for object detection, steadily rise over the course of the 490k iterations run to train the model.

Also, reading Figure 4, we can precisely conclude that over a set of different sized objects, large objects are easily detected due to their large area coverage, hence highest average precision. The small objects follow the exact opposite pattern in average precision.

When we consider all sized objects for precision over different area @ IoU from Figure 5, the 50% area overlap gives the maximum precision, while 75% area overlap does not, as one would think it would. The average precision of entire model is recorded to be 26.2% (matching with findings in [6]) and this parameter holds value as 29.3% for person, which is 0.1% less than the recorded precision in [6]. Also, when comparing the value of the average precision for each category with that of the model, nearly all the values are in range of precision 26.2% ±10, which draws the conclusion that the model has become specifically more accurate for the six categories.

6. SCOPE

However, the results do build paths to discover. The further work that can be accomplished is by reducing the number of iterations from 490k to, for example, 100k to measure whether the precision remains same or not. Also, the effective change in the precision can be checked by changing the number of objects of the subset used to fine-tune the model, while keeping the iterations as 490k. Another direction can be uncovered by checking the extent of the precision change by incorporating both of the above-mentioned changes.

ACKNOWLEDGEMENTS

The authors wish to thank Dr. M. B. Chandak, Professor and Head of the Department, Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur, for his continuous encouragement to pursue this research. Also, to accomplish this research, the authors immensely appreciate the guidance and constant assistance provided by Professor Neha Tirpude, Supervisor and Assistant Professor, Computer Science and Engineering, Shri Ramdeobaba College of Engineering and Management, Nagpur.

REFERENCES

[1] S. Kamate and N. Yilmazer, "Application of Object Detection and Tracking Techniques for Unmanned Aerial Vehicles," *Procedia Computer Science*, vol. 61, pp. 436-441F, 2015.
[2] D. Sonntag, *et al.*, "Overview of the CPS for Smart Factories Project: Deep Learning, Knowledge Acquisition, Anomaly Detection and Intelligent User Interfaces," *Springer International Publishing*, Cham, pp. 487-504, 2017.
[3] N. Becherer, *et al.*, "Improving optimization of convolutional neural networks through parameter fine-tuning," Springer International Publishing, London, 2017.
[4] Yosinski J., *et al.*, "How transferable are features in deep neural networks?" *Advances in neural information processing systems (Proceedings NIPS)*, vol. 27, pp. 1-9, 2014.
[5] R. Girshick, *et al.*, "Rich feature hierarchies for accurate object detection and semantic segmentation," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 580-587, 2014.
[6] D. Sonntag, *et al.*, "Fine-tuning deep CNN models on specific MS COCO categories," *Computing Research Repository, Computer Vision and Pattern Recognition*, eprint arXiv: 1709.01476, 2017.
[7] S. Ren, *et al.*, "Faster R-CNN: Towards real-time object detection with region proposal networks," *Advances in neural information processing systems*, pp. 91-99, 2015.
[8] T. Y. Lin, *et al.*, "Microsoft COCO: Common objects in context," *European conference on computer vision*, pp. 740-755, 2014.
[9] R. Girshick, "Fast R-CNN," *IEEE International Conference on Computer Vision (ICCV)*, 2015.
[10] S. Palmer, *et al.*, "Canonical perspective and the perception of objects," *Attention and performance IX*, vol. 1, 1981.
[11] D. Desai, Blog; https://deshanadesai.github.io/notes/Evaluation-of-Results-using-Mean-Avg-Precision.

BIOGRAPHIES OF AUTHORS

Garima Devnani is a final-year student in the Department of Computer Science and Engineering at Shri Ramdeobaba College of Engineering and Management. She will be receiving a Bachelor's degree in Computer Science and Engineering from Rashtrasant Tukdoji Maharaj Nagpur University in Nagpur, Maharashtra, India. She has completed projects in the fields of Web and Android App Development, Natural Language Processing, Data Science, and Machine Learning. Her current research interests include Data Science, Machine Learning, and Natural Language Processing.

Ayush Jaiswal is a final-year student of the Department of Computer Science and Engineering at Shri Ramdeobaba College of Engineering and Management. He will be receiving a Bachelor's degree in Computer Science and Engineering from Rashtrasant Tukdoji Maharaj Nagpur University in Nagpur, Maharashtra, India. He has worked on projects in the fields of Data Science, and Machine Learning. His current research interests include Artificial Intelligence, and Machine Learning. He plans to pursue his Master of Technology in Computer Science in near future.

Roshni John is a final-year student in the Department of Computer Science and Engineering at Shri Ramdeobaba College of Engineering and Management. She will be receiving a Bachelor's degree in Computer Science and Engineering from Rashtrasant Tukdoji Maharaj Nagpur University in Nagpur, Maharashtra, India. She has completed projects in the fields of Business Intelligence, Data Science, Deep Learning and Cyber Securities. Her current research interests include Data Science, Deep Learning, and Web Securities.

Rajat Chaurasia is a final-year student in the Department of Computer Science and Engineering at Shri Ramdeobaba College of Engineering and Management. He will be receiving a Bachelor's degree in Computer Science and Engineering from Rashtrasant Tukdoji Maharaj Nagpur University in Nagpur, Maharashtra, India. He has worked on numerous projects in fields of Web Development and Design, Enterprise Application Development, Data Analytics, and Machine Learning. His current research interests include Graphics Designing, and Machine Learning. He plans to pursue his Master's degree in near future.

Neha Tirpude is an Assistant Professor in the Department of Computer Science and Engineering at Shri Ramdeobaba College of Engineering and Management. She received the Master of Technology and Bachelor of Engineering degree in Computer Science and Engineering in 2013 and 2010 respectively. Her current research and publication interests include bio-medical image processing, data mining and information security.