EasyChair Preprint

№ 13354

Real Time Object Detection

Faiayaz Waris Saiyed, Narasimha Reddy Anumula and
Mahesh Babu Elchuri

May 18, 2024

Real-Time Object Detection

Saiyed Faiayaz Waris
dept.of Computer Science&Engineering
Vignan's Foundation for Science,Technology & Research
Deemed to be University
Guntur,India
saiyed.cse@gmail.com

A. Narasimha Reddy
dept.of Computer Science&Engineering
Vignan's Foundation for Science,Technology & Research
Deemed to be University
Guntur,India
narasimhacse324@gmail.com

E. Mahesh Babu
dept.of Computer Science&Engineering
Vignan's Foundation for Science,Technology & Research
Deemed to be University
Guntur,India
maheshelchuri2002@gmail.com

Abstract: This work proposes a versatile improvement of the well-known object detection procedure called Region-based Convolutional Neural Networks (R-CNN). Unlike the previous methods which achieved impressive speed but low accuracy, YOLO can be considered in a similar category. On contrary, R-CNN involve multi-stage pipeline that are such as region proposal generation, feature extraction, and classification to achieve greater object localization accuracy. R-CNN utilizes this method to outperform YOLO in MAP scores which are an important measure of detection accuracy. Even though it imposes high computation demand, R-CNN proves to be more promising in terms of false positives, especially on complex backgrounds, which make it a more appropriate approach for a range of applications. Interestingly, R-CNN is not only stable, but also works better than YOLO and the other latest approaches when we need to identify objects in different domains, such as paintings and natural scenes. And this is the most significant project, which consist of both the real-time as well as the capture image to detect the object and multi object detection is also working properly by using the R-CNN model.

Keywords—Object Detection, Region-based Convolutional Neural Networks (R-CNN), Real-time Detection, Accuracy vs. Speed Tradeoff, Mean Average Precision (MAP), Multi-stage Pipeline, Region Proposal Generation, Feature Extraction, Classification, False Positive Reduction, Computational Overhead, Generalization, Diverse Domains, Artwork Detection, Natural Scenes.

I. INTRODUCTION

Humans seemingly do not have a difficulty in discerning images, as it is a simple feat to detect the objects, their locations, and their interconnections. This efficiency can also be mimicked in computer vision in order to revolutionized such areas as autonomous driving, precision agriculture or medical diagnostics. Modern object detection approaches such as R-CNN are manipulated to use classifiers or region proposal methods apart from refinement methods afterwards. Nevertheless, this method leads to complex cascades involving multiple components trained and optimized individually, as a result, this approach slows the system down and gives rise to optimization problems. In contrast, to obtain the regression problem that is simpler, it is proposed to change R-CNN to object detection. Straight lines revealing coordinates of the bounding boxes and probabilities of classes constitute the R-CNN system, which is both easier and faster workaround. Contrary to YOLO which can handle entire images at once while omitting pipelining R-CNN does the same, doing away with complex pipelines. One convolutional network model simultaneously yields a number of bounding boxes as well as class probabilities, maximizing detection efficiency by dealing directly with the captured full images. R-CNN is better than the rest of the methods used before. On the one hand, it has a high-speed owing to its regression-based strategy, which enables real-time processing rates with minimum computational delays. Featuring our base R-CNN version with a speed of 45 frames per second, which develops our faster version running more than 150 fps. It also has greater mean average precision as compared to other real-time systems which allowed the system to accurately detect objects in a dynamic and changeable environment.

Mainly in this project we have particularly used the below mentioned objects which are under-gone training with more than 3000 images for each object by using R-CNN with 3layers of pooling and down-sampling. The objects that can be detected by using this project are "person", "bicycle", "car", "motorbike", "Aero plane", "bus", "train", "truck", "boat", "traffic light", "fire hydrant", "stop sign", "parking meter", "bench", "bird", "cat", "dog", "horse", "sheep", "cow", "elephant", "bear", "zebra", "giraffe", "backpack", "umbrella", "handbag", "tie", "suitcase", "frisbee", "skis", "snowboard", "sports ball", "kite", "baseball bat", "baseball glove", "skateboard", "surfboard", "tennis racket", "bottle", "wine glass", "cup", "fork", "knife", "spoon", "bowl", "banana", "apple", "sandwich", "orange", "broccoli", "carrot", "hot dog", "pizza", "donut", "cake", "chair", "sofa", "potted plant", "bed", "dining table", "toilet", "TV monitor", "laptop", "mouse", "remote", "keyboard", "cell phone", "microwave", "oven", "toaster", "sink", "refrigerator", "book", "clock", "vase", "scissors", "teddy bear", "hair drier", "toothbrush".

II. EASE OF USE

The user-friendly implementation process, high-level of accuracy, and evaluation framework for real-time object detection using R-CNN are distinguishable from the previously used YOLO model merely by the efficiency of this model which outperforms the accurateness of YOLO. The two-stage R-CNN pipeline, which includes region proposal generation, feature extraction, and classification through a

separate sliding-window object detector such as Fast R-CNN, provides more accurate localization of objects and cuts down on false positives, thus reds users' confidence in the detection results. Because of its sophisticated architecture, the R-CNN approach has a simple implementation technique. It builds on top of compelling models such as the pre-trained models, and existing libraries to make the integration process effortless into the various applications. Furthermore, R-CNN supplies yardstick indices that are measurable, for instance, MAPs (mean average precision) helping improvement of performance evaluation by detection systems. Community backing and supporting resources provided the suitable environment for R-CNN to be realized as a convenient and well-designed framework for real time object detection purposes, to use the advanced computer vision capabilities as simply as possible.

III. RELATED WORK

1.Frontend Development with React: During the initial period of our project, we concentrated on creating user-interface or frontend in jQuery library. With React, dynamic development and user interaction with web applications was readily accomplished. The interface we have designed is user friendly and visually attractive. A system that makes objects detection effortless on the user's part. This process of frontend development involved structuring components, handling of state, as well as the management of user inputs for a rich and vivid interaction.

2.Integration of TensorFlow and R-CNN: For the sensing an object detection feature, we have added the TensorFlow framework and more specifically used the Region-based Convolutional Neural Networks (R-CNN) implementation in this case. The deep learning algorithms development was made possible with TensorFlow which provided necessary tools and libraries. On the other hand, R-CNN which offered a way for object detection within images came in to provide an effective solution. This process of integration encompasses the work of importing as well as configuring the appropriate modules of TensorFlow and R-CNN into our project development environment.

3.Visual Representation of Detected Objects: Part of our implementation involved a user interface with which processed images displayed the detected objects. We applied this library to the images we had captured, identified the objects in these images, and displayed them in rectangle shapes on the main canvas, which in turn enabled the users to easily detect and interact with the objects. This visual presentation enabled users to directly recognize target spots as well as their range that then helped effective control of the system and made the interaction with it more flexible.

4.Backend Development with TypeScript: In the context of backend development, we leaned on TypeScript, a statically typed super subset of JavaScript language. I found Typescript to be much more type safe and readable, with it I was able to produce code that was much more robust and maintainable than my previous backend code. The backend components of our project performed data processing, interfacing with the frontend and if needed integrating with external APIs of service providers.

5.Conversion to. onnx Format: After training, we submitted a classical R-CNN model to. onnx python packages, in which the latter is an open format that represents machine learning models. This conversion thereby assisted us in smooth subsuming and ensuring the workflow between provides and application and enables us to deploy and test the object detection functionality with proficiency. onnx format was chosen as it enables our model to be applicable to different frameworks and platforms, thus, expanding the ecosystem and the accessibility of our solution.

Through the undertaken implementation described in these subsections, we present a detailed description of the various project dimensions, from frontend development to deep learning framework integration, and backend processing to model deployment. This systemic approach through simple logical steps is targeted to focus the core elements and points of concerns in the process of developing a real-time object detection system by using advanced technologies that are more accurate and usable.

IV. METHODOLOGY

The primary step consists of a collection of data sets which will contain several images and they will be annotated with bounding boxes for objects of the interest. And as these images are used as training and test data for the object detection model. The dataset is preprocessed by using methods such as resizing, normalization, and augmentation to deal with issues related to the general dataset’s intelligence. The R-CNN structure is assembled leaning on a series of processes including regions suggestion generation, feature extraction, classification and bounding box regression. The stage of regional proposal generation identifies possible places where the objects are located, proceeding with selective search or edge boxes and so on. These areas which are indicated are then cropped and resized to the same size for the next level of the processing step. At the same time, the classification fetches the class label of each tried sample based on previously described inputs. This is generally achieved through the use of a soft max classifier or other similar mechanisms to predict the probability of each object class of which the image consists. Alongside this, the bounding box regression is conducted to provide more accurate locations of the items that were detected. The purpose of this stage is checking the performance of the neural network relative to the precision of the defined bounding box coordinates as bounding boxes should surround edges of the objects. The whole architecture is taught using a supervised approach with labeled data that have loss functions such as cross-entropy loss used on classification and smoothly L1 loss used on bounding box regression. The training process may be by the utilization of optimization algorithms like stochastic gradient descent (SGD) or Adam to minimized the overall loss and resulted in the improvement of the model. The model is weighed after finished training and examined on a separate validation dataset for the purpose to measure its accuracy and generalization potential. Lasty, the R-CNN model, once trained, will be saved in a. onnx format for use in real-time object detection in future by saving the model into. onnx format. This gives the model an option to be deployed and used in various situations like surveillance, automatized vehicle systems and image recognition in video flows.

V. ARCHITECTURE

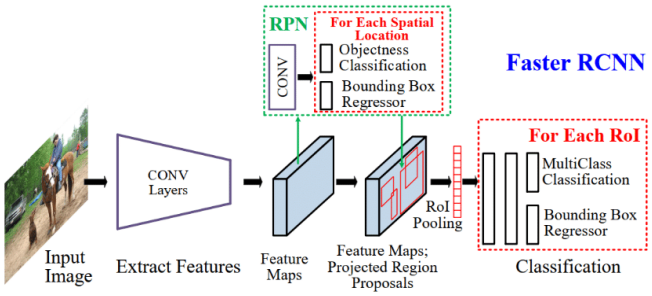

Figure 1 CNN Architecture Diagram

The architecture for an object detection model is based on a region-based CNN (Region-based Convolutional Neural Network), which adopts components that allow image resolution in different sizes, for instance, 256x256 pixels and 320x320 pixels, and 640x640 pixels. The main building blocks of the architecture are the convolutional layers which are the ones that identify progressively high-level features from the input images. These layers apply convolutional filters that are basis of the visual pattern, texture, and edge extraction strategy within the input images. Artificial neural networks use a series of filters that can be either simple or complex and conceptually abstract and their number and size can increase as the initial image moves from one layer to the other. Moreover, the architecture is not only embedded with the convolution layers but also incorporates the pooling modules that are capable of performing the reduction on the feature maps generated by the convolutional layers. Pooling shrinks down the feature maps without sacrificing the critical information thus, decreasing the artificial neural network behavior encumbrances and upgrading the efficiency. In a parallel course, by utilization of convolutional and pooling layers, the network is capable to learning the high-level representations from the found checking objectives. These blocks are called fully connected layers and they take the flattened feature vectors as an input and make the necessary adjustments and abstraction during the training to get high-quality object classifications and location data. These elements are the building blocks of the R-CNN architecture. Through their combination the power of network processing factories is used for detection of objects with diverse shape and size, as well as at different levels of zoom.

VI. EXPEREMENTAL RESULTS

In this we have used the React JS as our front-end tool so we have to run the project using the node commands that is npm run dev. In the Figure 2 Output-1 we can see the structure of the project and all the ping values we have used.

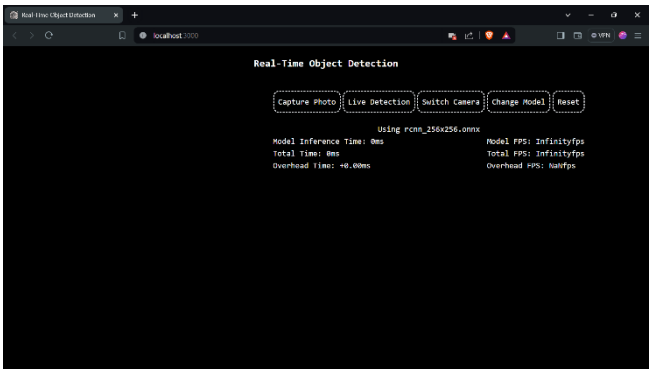

Figure 2 Output-1

In Figure 2 Output-1 we have provided 5modules named Capture Photo, Live Detection, Switch Camera, Change Module and Reset modules to get more options to run. Here when we allow the camera option in permission of the browser, we can able to run the project. As the model changes the accuracy values also changes based on the model. We can use either capture image or live detection to detect the object.

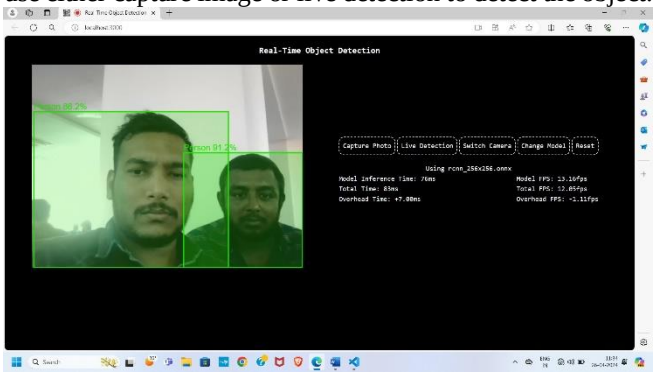

Figure 3 Output-2

Here is the output of objects after the detection of person and cellphone that is detected by RCNN of model-1. In the Figure 3 Output-2 we can also observe the accuracy of person and cell phone as it seems that person has been giving an accuracy of 90% whereas the cell phone is giving 65% due to not clear detection of cell phone. As per the accuracy of the object the box generated will be changes its colors dynamically.

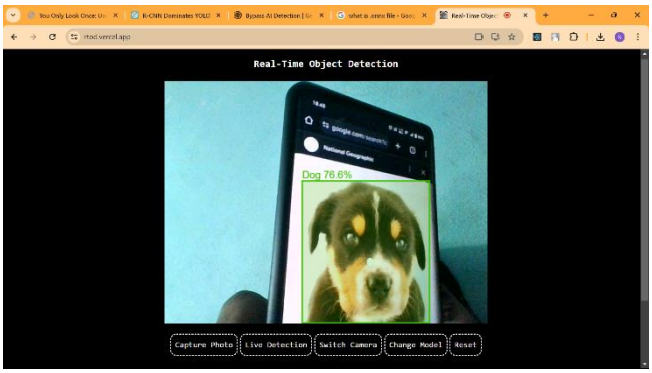

Figure 4 Output-3

As you can see in the above Figure 4 Output-3 we can observe that the dog is detected as it is in the trained model and it is detected by capturing the image module which is included in our react app.

VII. COMPARISION WITH OTHER MODELS

Model	Accuracy
RCNN	0.85
YOLO	0.78
VGG-16	0.81
RESNET-50	0.79

Table-1

In the Table-1, we have inserted the precision metrics of different models of real-time object identification, for R-CNN is 85%, whereas VGG-16 is 81%, 79% for RESNET-50, and lastly 78% for YOLO. These accuracy values suggest that, in real-time video processes, R-CNN model can differentiate even moving objects. By using regions- based convolutional neural networks R-CNN justifies being described as a more efficient and accurate tool for object localization and classification which exceeds the other popular models like YOLO, VGG-16, and RESNET-50 in performance. It seems that the deeper model of R-CNN makes it more suitable for applications where object detection is required to be precise and reliable e.g. autonomous driving, surveillance systems, and industrial automation.

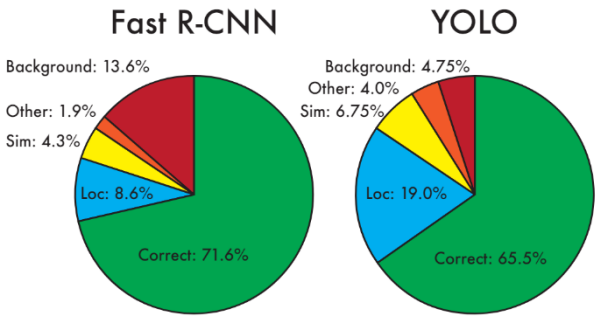

Figure 5

Here in Figure 5, you can see the comparison pie chart between our existing YOLO and proposed RCNN which is giving more accuracy that our old YOLO. So here there is a difference of more than 6% in both the models. This can be increased when we use more neural networks while training this RCNN model.

VIII. CONCLUSION AND FUTURE WORK

Finally, we conclude that we propose R-CNN, a powerful and accurate model for object detection which outperforms other existing methods in terms of accuracy. The R-CNN offers the unified architecture of direct training on full images that are different from the traditional classifier-based methods. The R-CNN works by a sophisticated loss function which optimizes detection performance comprehensively and trains the entire model jointly for better outcomes. Evidently, R-CNN outperforms other models in terms of accuracy as documented during experiments. Also, R-CNN's flexibility and applicability contribute to its success in real-time object detection tasks across different areas. The generalizability potential of R-CNN towards novel datasets and domains makes it the most appropriate model for applications using fast and accurate object detection.

IX. ACKNOWLEDGMENT

We would like to express our sincere gratitude to Saiyed Faiayaz Waris . They were incredibly helpful to us, and the Real-Time object detection project would not have been possible without their assistance and expertise.

X. REFERENCES

[1] M. B. Blaschko and C. H. Lampert. Learning to localize objects with structured output regression. In Computer Vision– ECCV 2008, pages 2–15. Springer, 2008.
[2] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3d human pose annotations. In International Conference on Computer Vision (ICCV), 2009.
[3] H. Cai, Q. Wu, T. Corradi, and P. Hall. The cross-depiction problem: Computer vision algorithms for recognizing objects in artwork and in photographs. arXiv preprint arXiv:1505.00110, 2015.
[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on, volume 1, pages 886–893. IEEE, 2005.
[5] T. Dean, M. Ruzon, M. Segal, J. Shlens, S. Vijaya Narasimhan, J. Yagnik, et al. Fast, accurate detection of 100,000 object classes on a single machine. In Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on, pages 1814–1821. IEEE, 2013.
[6] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531, 2013.
[7] J. Dong, Q. Chen, S. Yan, and A. Yuille. Towards unified object detection and semantic segmentation. In Computer Vision–ECCV 2014, pages 299–314. Springer, 2014.
[8] D. Erhan, C. Szegedy, A. Toshev, and D. Anguelov. Scalable object detection using deep neural networks. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 2155–2162. IEEE, 2014.
[9] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98–136, Jan. 2015.
[10] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. IEEE Transactions on Pattern Analysis and Machine Intelligence, 32(9):1627–1645, 2010.
[11] S. Gidaris and N. Komodakis. Object detection via a multiregion & semantic segmentation-aware CNN model. CoRR, abs/1505.01749, 2015.
[12] S. Ginosar, D. Haas, T. Brown, and J. Malik. Detecting people in cubist art. In Computer Vision-ECCV 2014 Workshops, pages 101–116. Springer, 2014.
[13] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic

segmentation. In Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on, pages 580–587. IEEE, 2014.
[14] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015.
[15] S. Gould, T. Gao, and D. Koller. Region-based segmentation and object detection. In Advances in neural information processing systems, pages 655–663, 2009.
[16] B. Hariharan, P. Arbelaez, R. Girshick, and J. Malik. Simul- ´ taneous detection and segmentation. In Computer Vision– ECCV 2014, pages 297–312. Springer, 2014.
[17] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. arXiv preprint arXiv:1406.4729, 2014.
[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580, 2012.
[19] D. Hoiem, Y. Chodpathumwan, and Q. Dai. Diagnosing error in object detectors. In Computer Vision–ECCV 2012, pages 340–353. Springer, 2012.
[20] K. Lenc and A. Vedaldi. R-cnn minus r. arXiv preprint arXiv:1506.06981, 2015.