Learning Without Forgetting in Object Detection

Projna Saha
Department of Computer Science
Carleton University
Ottawa, ON K1S 5B6
`projnasaha@cmail.carleton.ca`

Abstract

Convolutional Neural Network (CNN) is a powerful tool for object detection applications. While image classifications (e.g. CNN stacks) worked really well to classify images, object detections took one step further, drawing one or many bounding boxes of objects of interest. We propose a new approach using the Learning without Forgetting technique on the object detection model, which uses only new task data to train the network while preserving the original capabilities. Our method performs favorably compared to the Learning Without Forgetting (LwF) model.

Keywords: *Object detection, Faster-RCNN, Learning Without Forgetting, Computer Vision*

1 Introduction

Object detection is the process of identifying objects, scenes, or people in an image with a computer vision framework. The application involves tracking objects, video surveillance, pedestrian detection, anomaly detection, people Counting, self-driving cars, or face detection. Object detection technology has been a subject of much research and development due to the increasing use of images and videos as data sources and their huge number of applications [1]. The high-level goal of object detection is to automate the process of recognizing objects from images that are sampled from scenes.

A major challenge for object detection is the limited amount of annotated data currently available for machine learning models. Object detection datasets [2] often contain ground truth examples for about a dozen to a hundred classes of objects, while image classification datasets [3] can include upwards of 100,000 classes. It requires more computational power to train an object detection model. Based on the problem of detecting objects, it is very important to use the previously trained model [4] to fine-tune for the new task without degrading the performance on old tasks (Catastrophic Forgetting) [5][6].

In Learning Without Forgetting(LwF) [7], the authors addressed the problem of adapting a vision system to a new task while preserving performance on original tasks. In their work [7], authors demonstrated the efficacy of LwF for image classification problems that can make a considerable difference in terms of performance on both training and testing sets for image classification. The authors mentioned in their future work that they would like to do a further experiment on semantic segmentation, detection, and problems outside of computer vision. From this concept of their future work, we applied the LwF on object detection techniques to evaluate the performance of their model. This paper focuses on two research findings with regard to the object detection problem.

- RQ1: In the object detection problem, how well a model can learn for a new task on a new dataset if modifying the old model with a new task?
- RQ2: How do the results differ from the authors' findings?

In this experiment, we used a pretrained object detection model called Faster-RCNN [8] as an old model which was trained on COCO dataset[1]. Then we created a new model by adding a new task on top of the old model. Then we trained our new model with PennFudan Dataset [2]. We evaluated our model using COCO evaluation metrics 1 for object detection. Finally, we compare the old and new models using the PennFudan dataset after training and show their results. These results demonstrate that the old model retains its LwF knowledge and that the new model learns from prior information more effectively than the old model. In our experiment, the old model always underperforms than new model like LwF.

Our paper is organized as follows: Sec.2 briefly reviews the object detection techniques, as well as the reference paper for this work related to image classification technique; Sec.3 introduces dataset preprocessing, the description of our methodology Sec.4 describes the experimental setup & implementation details Sec.5 details the evaluation of our experiments and discussion; and Sec.6 concludes our work and future scope. Code is available at https://github.com/projna/Learning-Without-Forgetting-in-Object-Detection

2 Related Work

Detecting objects in images remains a challenging task for computer vision, especially in real-time applications. Despite the fact that there is the problem of adapting a vision system to a new task while preserving performance on original tasks, without access to training data for the original tasks. A group of researchers proposed the LwF method where a model can learn from the new tasks without having access to the old task data.

CNN-based object detection network has been briefly introduced in this chapter, maintaining that CNN-based object detection networks can be roughly divided into two ways: two-stage and one-stage. The two-stage object detection network usually has steps: the first step is to generate several proposals bounding boxes; the second step is to perform object classification and location information prediction on the proposal bounding boxes, its representative networks are R-CNN [9], Fast R-CNN [10], Region-Based Convolutional Neural Networks [11]), SPPNet [12], Faster R-CNN [8], FPN [13], Mask-RCNN [14] and YOLO [15], Faster R-CNN [8]. The Region-Based Convolutional Neural Networks or R-CNNs are in a family of machine learning models for computer vision and specifically object detection, while YOLO is part of the single-shot detector family. Both one-stage and two-stage approaches have advantages and disadvantages: we will explore four prevalent methods for object detection.

2.1 R-CNN

R-CNN is a state-of-the-art CNN architecture that has been applied to many problems in Computer Vision. R-CNN proposes [9] selective search based on region proposals and combines it with a greedy algorithm to generate feature maps. The final output of this architecture is a classification model which can be fed into a Support Vector Machine (SVM) to classify images. However, there are many problems with this algorithm. Training a R-CNN model takes a long time, and it does not generalize well because it is only trained on warped images. There may also be generations of bad candidates due to the selection process used in R-CNN's training algorithm.

2.2 Fast R-CNN

Fast R-CNN is a neural network that learns to predict object categories and bounding boxes. In the paper [10], authors showed an improvement of R-CNN which uses only one region proposal network with a goal of improving performance on challenging images by applying multiple CNN classifiers simultaneously. The Fast R-CNN is faster than the R-CNN because it shares computations across multiple proposals. Despite its advantages, the Fast R-CNN model depends on the time-consuming Selective Search algorithm, which cannot be customized.

[1]COCO Dataset: $https://cocodataset.org/\#download$

[2]PennFudan dataset: $https://www.cis.upenn.edu/\ jshi/ped_html/PennFudanPed.zip$

2.3 YOLO

YOLO (You Only Look Once) [15] is an object detection algorithm based on the bounding boxes from an image. It works by splitting the image into an SxS grid, within each of the grids we take m bounding boxes. YOLO is orders of magnitude faster than other object detection algorithms. The YOLO algorithm is limited by its capability to understand small objects in an image. For example, a flock of birds would be difficult for the algorithm to detect because they are spatially dense and there are multiple of them.

2.4 Faster RCNN

Faster R-CNN allows for better accuracy and faster performance, which translates to an easier time building any convolutional networks. Faster R-CNN [8] uses a new layer called ROI Pooling to extract equal-length feature vectors from all proposals in the same image and does not need as much disk storage as R-CNN. Faster R-CNN is a combination of RPN and fast R-CNN. Combining deep convolutional networks with recurrent neural networks, Faster R-CNN improves accuracy by leveraging the higher speed of RCNNs on image recognition. In our current work, We have used COCO dataset as the old model which was trained on Faster R-CNN.

2.5 Learning Without Forgetting (LwF)

Learning without Forgetting (LwF) [7] is an effective learning algorithm for neural networks that can learn a network that can perform well on both old tasks and new tasks when only new-task data is available. In the LwF Method, authors address the problem of adapting a vision system to a new task while preserving performance on original tasks, without access to training data for the original tasks. In their experiment, they demonstrated the effectiveness of LwF for image classification and one experiment on track, but in future work, they mentioned doing more experiments on semantic segmentation, and detection.

3 Methodology

3.1 Dataset

Pennfudan dataset: For our new task training, we have used the Pennfudan dataset which is an image database containing images that are used for pedestrian detection in the experiments reported in [16]. Each image has at least one pedestrian in it. The heights of labeled pedestrians fall into [180,390] pixels. The images are taken from scenes around campus and urban streets. There are 170 images with 345 labeled pedestrians among which 96 images are taken from around the University of Pennsylvania, and the other 74 are taken from around Fudan University. This dataset includes annotation of the images, original image, and mask and in the annotation of the image, it contains the object bounding box information, pixel mask, image size, number of objects, and image name. For the mask, all the backgrounds are labeled as '0' zero, which means the background is black. Also, it contains two objects, one is the person and another is a background.

COCO Dataset: Our old model is trained on the COCO dataset which is a large-scale dataset for objection detection, segmentation, and captioning. This dataset provides two tasks one is the bounding box task and the other is the segmentation task. The COCO dataset contains almost 330,000 images and approximately 80 object categories and 91 stuff categories. It also includes a complete segmentation mask for every object in the dataset. All of the annotations and images are publicly available for research purposes.

3.2 Dataset Preprocessing

At the beginning of the data preprocessing, we align all the images. Then we convert the image to RGB without converting the mask to RGB. Then we removed the background from the mask. Because we only want to know the features of the object. Then, we collect bounding box coordinates from the annotation folder for each of the masks. Then we labeled all the target folders. The target folder contains boxes, labels, masks, image IDs, areas,s and images crowded or not. For the training images, we transformed the image file into Tensor and also, randomize the horizontal flip with the value of 0.5.

3.3 Model Description

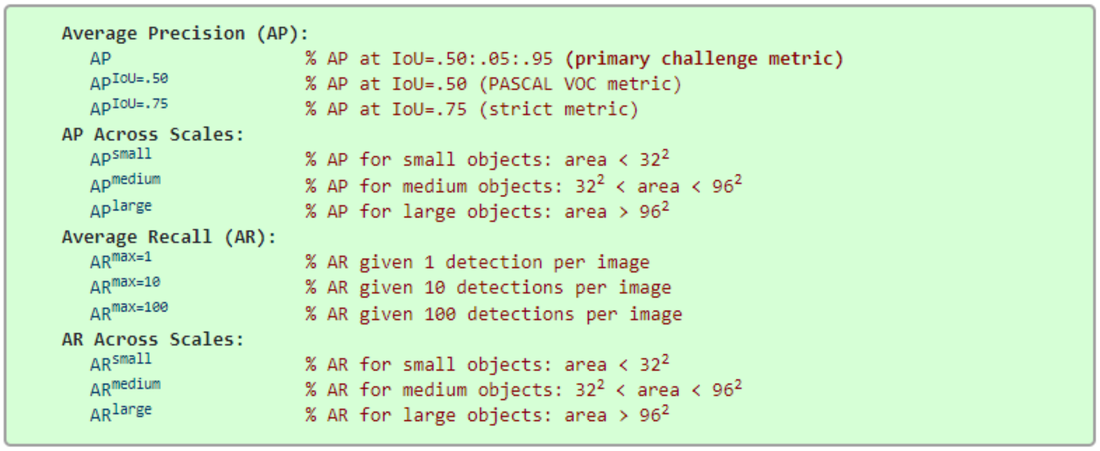
For the model, we have used state-of-the-art Faster R-CNN which is an improved version of R-CNN. The architecture of Faster R-CNN is structured into two units, the first is Region Proposal Network (RPN) and the second one is Fast R-CNN.

RPN is a fully convolutional neural network with the help of attention. It creates an aspect ratio of the image so that the model knows where to find the object from the given image. And Fast R-CNN is responsible for detecting the object with a given RPN score. In short, the Faster R-CNN works as follows:

The RPN score suggests which region to look at. With all the RPN scores, it creates a fixed-length feature vector using the ROI pooling layer and Fast R-CNN classifies the image with the help of extracted feature vector. In the next step, the class score of the detected object and bounding box are returned. The input size of Faster R-CNN is 1024 and the output size is 91. We have used a pretrained version of Faster R-CNN from torchvision which was trained on COCO dataset.

3.4 Evaluation Metrics

For evaluation metrics, we used two standard evaluation metrics for object detection. One is Mean Average Precision (MAP) and another is Mean Average Recall (MAR). We have calculated precision and recall over 12 Intersection Fig.1 in over Union (IoU).



Average Precision (AP):
- AP % AP at IoU=.50:.05:.95 **(primary challenge metric)**
- $AP^{IoU=.50}$ % AP at IoU=.50 (PASCAL VOC metric)
- $AP^{IoU=.75}$ % AP at IoU=.75 (strict metric)

AP Across Scales:
- AP^{small} % AP for small objects: $area < 32^2$
- AP^{medium} % AP for medium objects: $32^2 < area < 96^2$
- AP^{large} % AP for large objects: $area > 96^2$

Average Recall (AR):
- $AR^{max=1}$ % AR given 1 detection per image
- $AR^{max=10}$ % AR given 10 detections per image
- $AR^{max=100}$ % AR given 100 detections per image

AR Across Scales:
- AR^{small} % AR for small objects: $area < 32^2$
- AR^{medium} % AR for medium objects: $32^2 < area < 96^2$
- AR^{large} % AR for large objects: $area > 96^2$

Figure 1: The 12 metrics are used for characterizing the performance of an object detector on COCO Dataset [4]

4 Experiment

4.1 Experimental Setup

In the experimental setup, to control the loss function, we used SGD as an optimizer with a learning rate of 0.0005 and the momentum of the learning rate is 0.09 with a weight delay of 0.0005. In order to determine the appropriate learning rate, we also employed StepLR as a scheduler. As we have two models, we need to control the loss function. So, we have used the α value of 0.01. Following that, we run the model with 50 epochs. We trained the model with PennFudan Dataset.

[4]Evaluation Metrics: $https : //cocodataset.org/\#detection - eval$

4.2 Implementation Details

There are two ways to fine-tune the pretrained model. One is changing the last hidden layer and the other is changing the backbone of the last layer for faster prediction. Here, we have used the second approach which changed the last layer with ***'MaskRCNNPredictor'*** for computing the instance segmentation of the mask. In our old model, we used a pre-trained model of Faster R-CNN named ***'fasterrcn_resnet50_fpn'*** which can predict 91 classes. We modified our new model with two new classes from the PennFudan dataset. After that, we fine-tuned the last layer. We have used 93 classes for a new model and we set the default value for the old model.

Before starting the training, we freeze the old model parameters, so that it could not train with the new dataset. And we have used batched size of 8. For calculating the loss function, we have used the following equation 1,

$$loss = \alpha * new\ model + (1 - \alpha) * old\ model \tag{1}$$

As the old model is not trained with the new dataset, so the loss for the old model will be less than the new model. α is a loss balance weight, set to 1 for most of our experiments. Making α larger will favor the old task's performance over the new task's, so we can obtain an old-task-new-task performance line by changing α.

5 Results & Discussion

Loss Function: We have evaluated our results for both old task and new task performance. We have calculated three loss curves, loss for learning the mask Fig.2(a) the second one for loss classifier Fig.2(b), and the third one for total loss Fig.2(c). Generally, the lower the loss value better the model learns. From Fig.2(a). loss mask curve, we can see that from epochs 2 to 8 it has a smooth curve with a gradually decreasing order. The loss mask curve is the most stable curve out of the 3 curves. From Fig.2(b), it is our classifier loss curve, in the first epochs, it starts from the low and then increases in the second epochs after that it gradually decreases but in epoch 6 it increases again, and in the next epochs, it decreases again. The reason is that our trained model is not stable yet or our evaluation dataset is very small. From Fig.2(c) we can see that the loss curve is very unstable. The reason is the effect of old model loss also. we have calculated the loss function with both new model loss and old model loss and we have used a 0.01 alpha value for balancing.

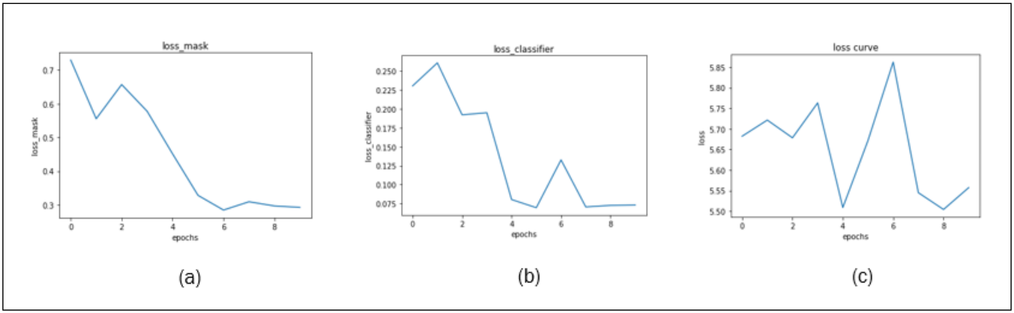
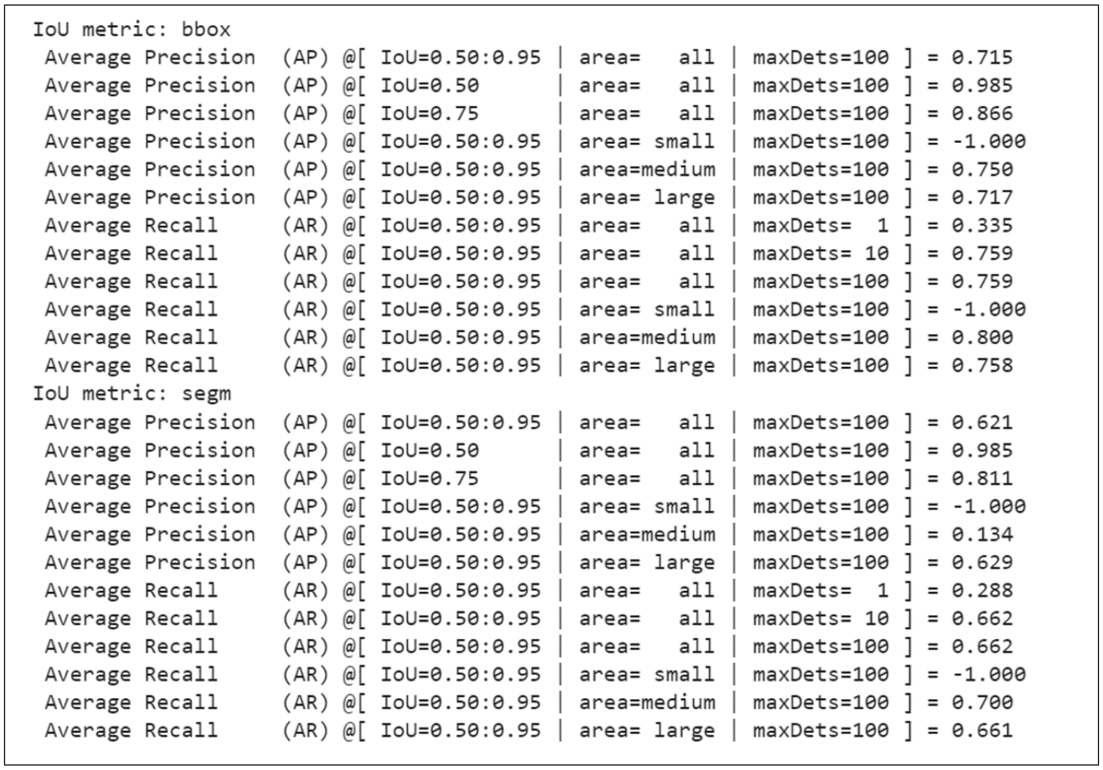


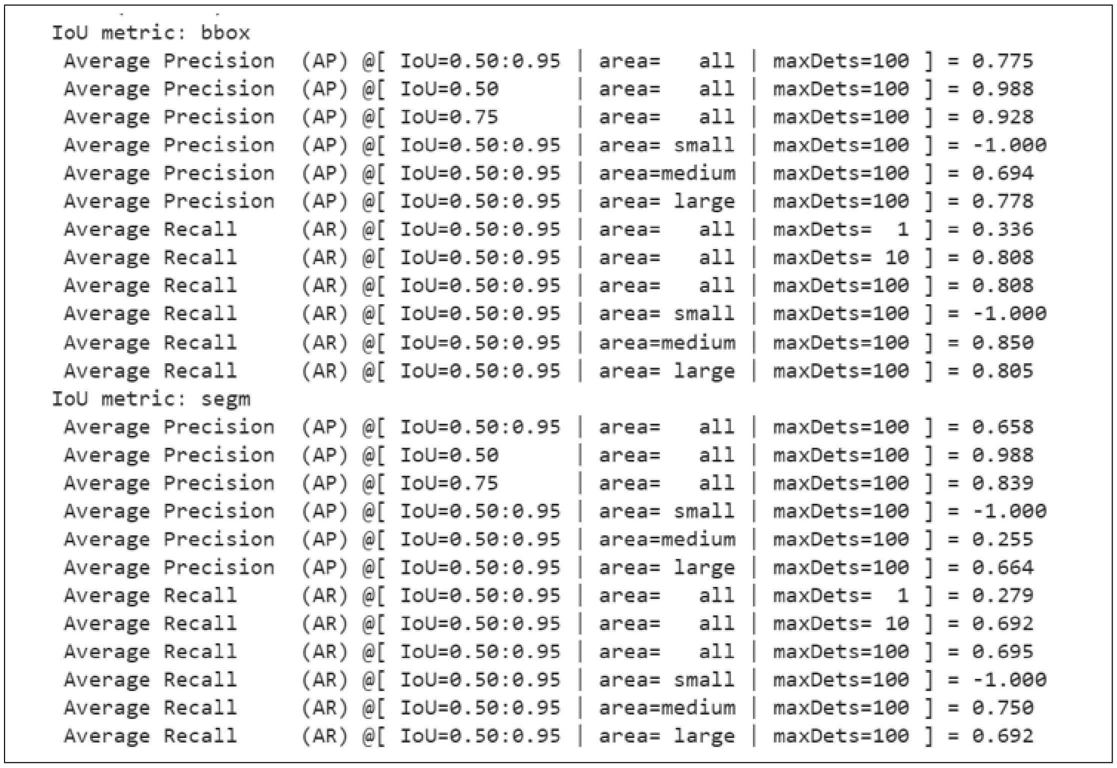
Figure 2: (left to right) (a) Loss/Localization Mask, (b) Loss Classifier, and (c) Loss Curve

Comparison between Old vs New Model: For model evaluation, we have used Fig.1 which shows 12 evaluation metrics from coco evaluation. We have also evaluated the result for both the bounding box and segmentation. How accurately the model can draw a box around the object is called a "bounding box test" and the Segmentation is how well the model can segment the object from the background. From Fig.3 which is our old model evaluation where we can see that all areas (0.50) highest precision of .985 and recall also .985. If we consider the area between .50 to .95, it has a precision of .715 and a recall of .521. For a medium area, it has a precision of .800 and .700. From Fig.4 which is our new model evaluation where we can see that all areas (0.50) have the highest precision of .988 and recall also .988. If we consider the area between .50 to .95, it has a precision of .775 and a recall of .658. for a medium area, it has a precision of .850 and .750. In Fig.5, we have compared both the old and new models based on mAP(mean Average Precision) and mAR(



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.715
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.985
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.866
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.750
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.717
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.335
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.759
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.759
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.800
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.758
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.621
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.985
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.811
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.134
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.629
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.288
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.662
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.662
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.700
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.661
```

Figure 3: Validation of Old Model



```
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.775
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.928
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.694
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.778
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.336
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.808
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.808
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.850
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.805
IoU metric: segm
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.658
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.988
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.839
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.255
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.664
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.279
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.692
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.695
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.750
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.692
```

Figure 4: Validation of New Model

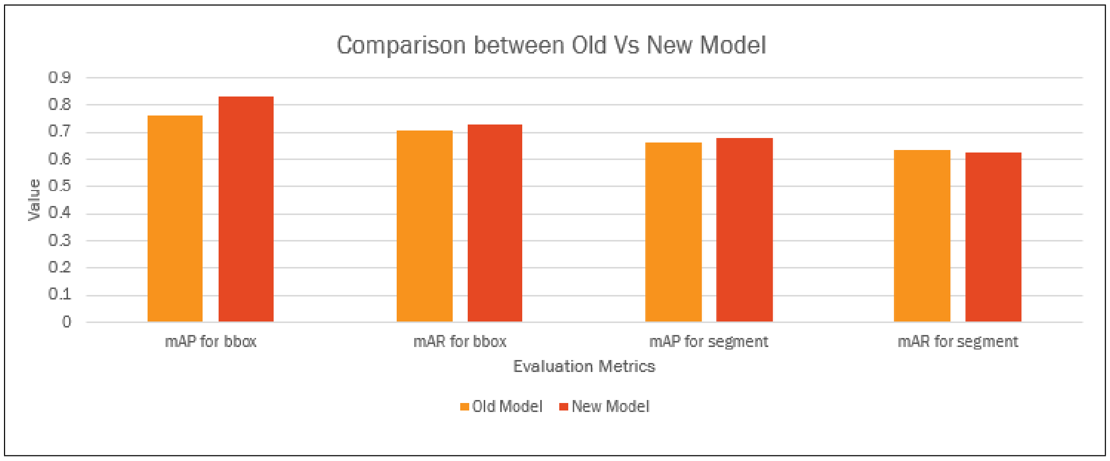


Figure 5: Comparison between Old vs New model

mean Average Recall). We can see that the new model performs better in all the metrics. We are also hoping that the old model will decrease some performance because it did not know the new model classes and did not train on the new dataset. According to LWF, the new model will always have a better result. From our comparison, we can definitively state that their model and our model behave in the same manner.

Comparison between Lwf in Image Classification & Our Approach: We compared our findings with the authors' findings in Table.1 and Table.2. As both of our evaluation metrics are different from each other, that is why we are comparing our results based on the percentage change between the old and new model. The authors used traditional classification evaluation metrics which is "Accuracy". We have used the most common evaluation metrics for object detection which are "mAP" and "mAR". For the authors' model, we can see that the old model degrades its performance by almost 15% on average. We can also observe that some of the experiments old degrade very little like 3%. From our result, we can conclude that the old model degrades by almost 4%. If we compare both models it can be observed that the object detection old model performs comparatively very accurately than the classifications' old model. The reason is that for object detection model has more features provided for a single image compared to image classification. Old-model object detection may therefore retain a lot more data than old-model classification. In this way, our results diverge from the authors' results.

Table 1: Author's LwF

Datasets	Old Model Accuracy	New Model Accuracy
ImageNet $\rightarrow$ VOC	56.2	76.1
ImageNet $\rightarrow$ CUB	54.7	57.7
ImageNet $\rightarrow$ Scenes	55.9	64.5
Places365 $\rightarrow$ VOC	50.6	70.2

Table 2: Our's LwF

Datasets	Old Model Accuracy	New Model Accuracy
mAP for bbox	76.18	83.26
mAR for bbox	70.58	73.04
mAP for segment	66.02	68.08
mAR for segment	63.42	62.16

Result of Object Detection: In Fig.6, it contains 6 persons or objects in the images. If our models can detect every object in the image we can say that our object detection models are performing better. Fig.7 proves that it can successfully detect all 6 objects in the image and label them as persons. So our object detection model is accurate as our results also showed that in above.

Figure 6: Reference Image

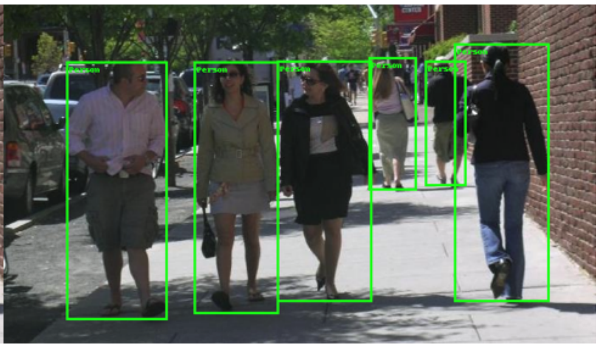


Figure 7: Predicted object with bounding box

6 Conclusion & Future Scope

We address the problem of adopting a vision system to a new task while preserving performance on original tasks, without access to training data for the original tasks. In this experiment, We have implemented a method in the context of object detection using the Learning Without Forgetting technique to determine how well a model can learn for a new task on a new dataset if modifying an old model is with a new task. And we also made a comparative analysis between our model's object detection using Lwf vs the author's Lwf for the Image classification problem.

The result showed that object detection can retain the old knowledge which is acting the same as LwF and also it detects the object effectively. And it always underperforms old models rather than new models which indicates the same result as LwF.

Not only is that a potential drawback of our approach but it's also restricting us from exploring other areas/overcrowded images for training the model. Object detection is a major issue in deep learning, where more than 80% of the processing time is spent on object recognition. This model was trained on the PennFudan dataset and consists of person and background classes. It's only trained in these two classes but not in any other category. The limited amount of annotated data currently available for object detection proves to be another substantial hurdle that must be overcome.

In the future, research may concentrate more on choosing datasets so that the model can create categories to characterize the objects, such as a crowd of people or a lot of traffic. The technology might be extended by working on motion pictures or videos to track the item. Faster RCNN has problems with detecting objects if there are a lot of crowds. YOLOv7 is the most accurate and fastest technique for object detection[8] but YOLOv7 is upcoming soon on PyTorch.

References

[1] Narayana Darapaneni, Sunilkumar C M, Mukul Paroha, Anwesh Reddy Paduri, Rohit George Mathew, Namith Maroli, and Rohit Eknath Sawant. Object detection of furniture and home goods using advanced computer vision. In *2022 Interdisciplinary Research in Technology and Management (IRTM)*, pages 1–5, 2022. doi: 10.1109/IRTM54583.2022.9791508.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[3] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John M. Winn, and Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111:98–136, 2014.

[4] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, Wentao Han, Minlie Huang, Qin Jin, Yanyan Lan, Yang Liu, Zhiyuan Liu, Zhiwu Lu, Xipeng Qiu, Ruihua Song, Jie Tang, Ji-Rong Wen, Jinhui Yuan, Wayne Xin Zhao, and Jun Zhu. Pre-trained models: Past, present and future. *AI Open*, 2:

225–250, 2021. ISSN 2666-6510. doi: https://doi.org/10.1016/j.aiopen.2021.08.002. URL https://www.sciencedirect.com/science/article/pii/S2666651021000231.

[5] Michael McCloskey and Neal J. Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of Learning and Motivation - Advances in Research and Theory*, 24(C):109–165, January 1989. ISSN 0079-7421. doi: 10.1016/S0079-7421(08) 60536-8. Funding Information: The research reported in this chapter was supported by NIH grant NS21047 to Michael McCloskey, and by a grant from the Sloan Foundation to Neal Cohen. We thank Sean Purcell and Andrew Olson for assistance in generating the figures, and Alfonso Caramazza, Walter Harley, Paul Macaruso, Jay McClelland, Andrew Olson, Brenda Rapp, Roger Rat-cliff, David Rumelhart, and Terry Sejnowski for helpful discussions.

[6] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013. URL https://arxiv.org/abs/1312.6211.

[7] Zhizhong Li and Derek Hoiem. Learning without forgetting, 2016. URL https://arxiv.org/abs/1606.09282.

[8] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf.

[9] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. doi: 10.1109/CVPR.2014.81.

[10] Ross Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. doi: 10.1109/ICCV.2015.169.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Region-based convolutional networks for accurate object detection and segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 38(1):142–158, 2016. doi: 10.1109/TPAMI.2015.2437384.

[12] Peijian Qu, Nan Liu, Zhengpeng Qin, Tianbo Jin, Hongze Fu, Zihao Li, and Peisheng Sang. Deep learning based detection of plant nutrient deficiency symptom and design of multi-layer greenhouse system. In *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, page 641–645. IEEE Press, 2022. doi: 10.1109/ICMA54519.2022.9856335. URL https://doi.org/10.1109/ICMA54519.2022.9856335.

[13] Jun Wang, Qiujuan Tong, and Chan He. A longitudinal dense feature pyramid network for object detection. In *2021 4th International Conference on Artificial Intelligence and Pattern Recognition*, AIPR 2021, page 518–523, New York, NY, USA, 2022. Association for Computing Machinery. ISBN 9781450384087. doi: 10.1145/3488933.3488939. URL https://doi.org/10.1145/3488933.3488939.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2017. URL https://arxiv.org/abs/1703.06870.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2015. URL https://arxiv.org/abs/1506.02640.

[16] Liming Wang, Jianbo Shi, Gang Song, and I-fan Shen. Object detection combining recognition and segmentation. In Yasushi Yagi, Sing Bing Kang, In So Kweon, and Hongbin Zha, editors, *Computer Vision – ACCV 2007*, pages 189–199, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-76386-4.