Analysis And Comparison Of Two Object Detection Models

Sheng Gao,Junxi Xu,Henghao Zhang
University of Toronto
{sheng.gao,stuart.xu,finn.zhang}@mail.utoronto.ca

Abstract

The primary objective of this final project is to perform exploratory research on two algorithms in the field of computer vision and on the subject of object detection and segmentation: **YOLO v4 and the Mask R-CNN algorithm**. The main task here is to reproduce the experimental results from existing papers, perform sensitivity analysis on hyper-parameters, and extend existing algorithms to the new dataset. The former algorithm is tested with two different feature extraction backbones with a single GPU on AWS, and the effects on the performance of the object detection models are justified. As for the latter one, the Mask R-CNN model was pretrained first and was used as the base model to see if there are any object detection tasks that Faster R-CNN fails to perform but the YOLO v4 may improve.

1 Introduction

Object detection and segmentation are computer technology related to computer vision and image processing that deals with detecting objects of different classes. For instance, humans, city views, buildings, cars, digital images, and videos. Also, the vision community has rapidly improved object detection and semantic segmentation results in the past year. Nevertheless, the majority of CNN-based object detectors are largely applicable only to recommendation systems.

In principle, Mask R-CNN[2] is an extension of Faster R-CNN[3] by adding a branch for predicting segmentation masks on each Region of Interest (RoI), in parallel with the existing branch for classification and bounding-box regression. The mask branch is a small fully convolutional network applied to each RoI and predicts a segmentation mask in a pixel-to-pixel manner. Given the Faster R-CNN framework, Mask R-CNN is simple to apply and train and it facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation. Most importantly, Faster R-CNN was not designed for pixel-to-pixel alignment between network inputs and outputs. To solve the problem, Mask R-CNN proposes a simple, quantization-free layer called RoIAlign, which preserves exact spatial locations.

As a result, the YOLO v4 and the Mask R-CNN algorithms are used to address these issues and check their respective effectiveness for resolving issues related to object detection.

2 Related work

Object detection relies not only on recognition accuracy but also on localization precision. Researchers in this area have put much effort into developing high-accuracy and high-efficiency deep learning algorithms. For this project, we exhibit two different state-of-art algorithms, which have tremendous improvement in both run speed and recognition accuracy in the field of object detection during the recent year.

In terms of object detection, the Region-based CNN (R-CNN) is primarily related work for Mask R-CNN. The way R-CNN approaches bounding-box object detection is to attend to a manageable

number of candidate object regions and evaluate convolutional networks independently on each RoI. R-CNN was extended to allow attending to RoIs on feature maps using RoIPool, leading to fast speed and better accuracy. Faster R-CNN advanced this method by learning the attention mechanism with a Region Proposal Network (RPN). Also, Faster R-CNN is flexible and robust to many follow-up improvements and is used as the benchmark for several current leading frameworks.

Previously, detectors consist of the head and backbone. The head part is used to predict classes and bounding boxes of objects, and the backbone is pre-trained on ImageNet. As for now, YOLO 4 adds a neck of an object detector between the backbone and head. There are many bottom-up paths and many top-down paths within a neck of an object detector. Also, for detectors running on the GPU platform, their backbone can be in various forms. For instance, VGG, ResNet, ResNeXt, and DenseNet. For those detectors running on the CPU platform, their backbone could be SqueezeNet, MobileNet, or ShuffleNet. As to the head part, it is usually categorized into two kinds, one-stage object detector, and two-stage object detector. The most representative two-stage object detector is the R-CNN series, including fast R-CNN, Faster R-CNN, R-FCN, and Libra R-CNN. It is also likely to make a two-stage object detector an anchor-free object detector like RepPoints. As for the one-stage object detector, the most representative models are YOLO, SSD, and RetinaNet. In recent years, anchor-free one-stage object detectors are developed. The detectors of this kind are CenterNet, CornerNet, and FCOS. Object detectors developed in recent years often insert some layers between the backbone and head, and these layers are usually used to collect feature maps from different stages.

In general, these two different deep algorithms achieve remarkable recognition accuracy and also have a high efficiency on training speed.

3 Detailed Method and Algorithm

YOLO v4: YOLO v4 utilizes a variety of prominent techniques in computer vision to improve the accuracy and optimize training speed for object detection. The primary objective for YOLO v4 is to achieve fast operating for parallel computation on a single GPU with high accuracy. To make this architecture more efficient, CSPDarknet53 is selected as the backbone to increase the receptive field size as well as the number of parameters. Also, SPP and PAN modules are used as the neck. SPP integrates Spatial Pyramid Matching (SPM) into CNN using max-pooling to significantly increase the receptive field with almost no reduction of operation speed. PAN refers to Path Aggregation Network which is another technique to increase the receptive field. Finally, the state-of-the-art YOLO v3 anchor-based head is used for dense prediction. Other modern and universal "tricks" are used to improve the model performance. “Bag of freebies” refers to methods that improve accuracy without increasing inference cost. Such techniques used in YOLO v4 include CutMix Mosaic data augmentation, DropBlock regularization, Cross mini-Batch Normalization (CmBN), CIoU-loss, Self-Adversarial Training, Optimal hyperparameters, etc. Similarly, “Bag of specials” refers to methods that only increase tiny inference costs to achieve noteworthy accuracy. YOLO v4 uses Mish activation, Cross-stage partial connection, Multi-input weighted residual connections, and modified-Spatial Attention Module (SAM) block as the “Bag of specials”.

Mask R-CNN: Mask R-CNN is theoretically simple to apply since it builds on Faster R-CNN. Faster R-CNN has two outputs for each candidate object, a class label, and a bounding-box offset, and for Mask R-CNN the only missing piece is a third branch which outputs the object mask. However, this additional mask output is distinct from the class and box outputs and requires extraction of the finer spatial layout of the desired object. The key elements of the Mask R-CNN model, including pixel-to-pixel alignment, are what differentiates Mask R-CNN and Faster R-CNN. Faster R-CNN consists of two stages. The first stage, Region Proposal Network (RPN), proposes candidate object bounding-boxes. The second stage, which is a Fast R-CNN, extracts features using RoIPool from each candidate box and performs classification and bounding-box regression. These features used by both stages can be shared for faster inference. In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to other popular systems, where classification depends on mask predictions. Lastly, for the Mask R-CNN model, the backbone architecture can also be evaluated using the ResNet or ResNeXt networks with a depth of 50 or 101 layers. The original implementation of Faster R-CNN with ResNets extracted features from the final convolutional layer of the fourth stage, which is called C4. Another more effective backbone used is a Feature Pyramid Network (FPN). FPN uses a top-down layout with lateral connections to build an in-network feature pyramid from a single-scale input. Faster R-CNN

Table 1: Mask R-CNN Results

Model	baseline AP	AP	AP@50	AP@75
R101-RPN	35.4	39.05	84.75	29.56

with an FPN backbone extracts RoI features from different levels of the feature pyramid according to their scale, but otherwise, the rest of the approach is similar to vanilla ResNet. As for the head part, it is extended from the Faster R-CNN box heads from the ResNet and FPN papers[3]. The head-on ResNet-C4 backbone includes the fifth stage of ResNet, which is computationally expensive. For FPN, the backbone already includes res5 and thus allows for a more efficient head that uses fewer filters.

4 Data Preparation, Experiments and Discussions

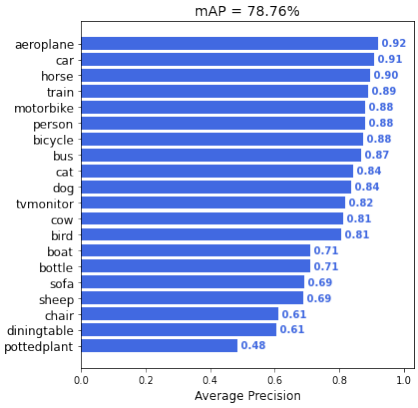
Data Preparation: Initially, both YOLO v4 and Mask R-CNN (RNN were pre-trained with a 3x schedule, which is about 37 COCO epochs) were pre-trained on the standard COCO dataset to obtain the pre-trained weights. Then, the YOLO v4 model was trained on the PASCAL VOC2007 dataset to see its effectiveness on images rather than COCO. Before analyzing the results from the YOLO v4 model, the prediction results from the Mask R-CNN model were used as the baseline and the detailed results can be found from Facebook AI Research (FAIR) [2][3]. To see how the performance of the Mask R-CNN model varies for the different datasets, the Mask R-CNN model was also trained on a wheat image dataset in COCO format from Kaggle. The wheat dataset contains images of wheat fields with bounding boxes for each identified wheat head, and there are 3035 training images and 338 validation images. The images were recorded in many locations around the world and not all images include wheat heads.

Experiment and Discussion for Mask R-CNN: For the Mask R-CNN model experiment, the object detection model was extracted from the Detectron2[4] library, which is a Facebook AI Research software system that implements state-of-the-art object detection algorithms. The selected architecture (with the best box AP), R101 - RPN, consists of a Feature Pyramid Network (FPN) as the backbone part and a standard header. This backbone uses a ResNet + FPN backbone with standard Conv and FC heads for mask and box prediction respectively, which obtains the best speed-accuracy tradeoff. Also, Mask RNN was initialized from backbones pretrained on ImageNet classification tasks, which is a converted copy of MSRA's original ResNet-101 model.

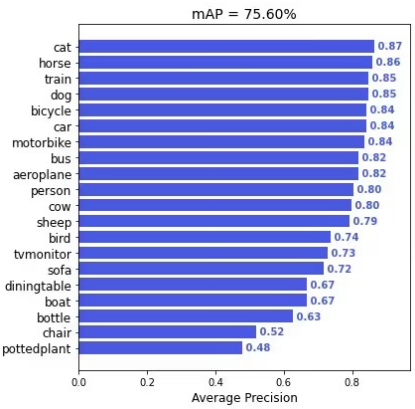
First, table 1 shows that the baseline box AP provided by Detectron2 and the results for the wheat dataset. Clearly, for the Mask-RNN model, the AP score on the wheat data outperforms the baseline score provided by the paper. There are several reasons for that. First, compared to the original large COCO dataset, which has more than 300k images and 80 object categories, the wheat data only has one class of wheat and there are 3035 images for training. For validation, 338 images were used and the targets are easy to distinguish. The pretrained model has covered a significant amount of data and extra training on the customized data helps improving the accuracy. Also, for the default Detectron2, it Uses scale augmentation during training. This improves AP with lower training costs. Likewise, it uses L1 loss instead of smooth L1 loss for simplicity, which sometimes improves box AP but may affect other AP.

Experiment and Discussion for YOLO v4: Compared 2 different backbones within the YOLO v4 model, which are CSPDarkNet53 and MobileNet v3. CSPDarkNet53 pretrained on COCO initially, and the YOLO head was used to predict classes and bounding boxes of objects. In our experiment, we freeze 50 epochs on backbone **Backbone: CSPDarkNet53** and only fine-tune on YOLO head with PASCAL VOC2007, which is a 20 classes object dataset. And next, we unfreeze extra 50 epochs on backbone CSPDarkNet53 and fine-tune on the entire YOLO v4 model. Finally, model achieved mAP: 78.76% (shown on Figure1:CSPDarkNet53mAP). **Backbone: MobileNet v3[5]** actually did the same thing, and finally achieved mAP: 75.60% (shown on Figure2:MobileNetv3mAP).

Discussion: YOLOv4 model with CSPDarkNet53 achieved a higher mAP value than that with MobileNet v3, but interestingly that using MobileNet v3 as backbone was able to detect more objects(Comparing pictures shown on Figure3:CSPDarkNet and Figure4:MobileNet). We also modified



(a) Figure1:CSPDarkNet53mAP



(b) Figure2:MobileNetv3mAP

(c) Figure3:CSPDarkNet

(d) Figure4:MobileNet

(e) Figure5:CSPDNetAdv

the CSPDarkNet53 with SAM and DropBlock structures. Compared with original CSPDarkNet53, this advanced CSP structure (Figure5:CSPDNetAdv) is successful to detect more objects with higher confidence.

5 Summary

In conclusion, both methodologies have been proved to be efficiently addressing certain challenges in real-world object detection. Comparing the YOLOv4 model with 2 different backbones, Mobilenet v3 as backbone seems more efficient than CSPDarkNet 53 due to computation cost and memory cost. Comparing the YOLOv4 model with Mask R-CNN, Mask R-CNN can detect small objects, but the effectiveness of Mask R-CNN is lower than YOLOv4.

References

[1] M. Bochkovskiy, C.-Y. M. Wang, and H.-Y. M. Liao, “YOLOv4: Optimal Speed and Accuracy of Object Detection.”

[2] He, K., Gkioxari, G., Dollar, P. and Girshick, R., 2018. Mask R-CNN. [online] Available at: <https://arxiv.org/pdf/1703.06870.pdf> [Accessed 19 April 2021].

[3]S.Ren, K.He, R. Girshick, and J. Sun, “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks.”

[4]Detectron2.readthedocs.io. 2021. Welcome to detectron2’s documentation! — detectron2 0.4 documentation. [online] Available at: <https://detectron2.readthedocs.io/en/latest/> [Accessed 20 April 2021].

[5]Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, Hartwig Adam, "Searching for MobileNetV3"

[6]code reference: https://github.com/bubbliiiing/yolov4-pytorch

[7]code reference: https://github.com/bubbliiiing/mobilenet-yolov4-lite-pytorch

Contributions

Sheng Gao: YOLOv4 with CSPDarkNet53 as the backbone, wrote 1 page of the report
Junxi Xu: YOLOv4 with MobileNet v3 and CSPDarkNet53 Advanced as the backbone, wrote 1 page of the report
Henghao Zhang: Mask R-CNN with R101-RPN as the backbone, wrote 2 pages of the pages