A Comparative Study of YOLO and DeepLabv3 for Object Detection and Image Segmentation Tasks

David Felipe Alvear Goyes

March 8, 2023

1 Introduction

Object detection and image segmentation are important tasks in computer vision with numerous applications, such as self-driving cars, image search, and medical image analysis. Two popular methods for these tasks are YOLO and DeepLabv3. In this analysis, we will discuss the contributions of each of these papers and compare the architectures and tricks used for object detection and image segmentation.

2 YOLO

2.1 Introduction

The YOLO paper [1] proposes an end-to-end method for object detection, which treats the problem as a regression task by predicting not only the class probability but also the bounding boxes. Overall, this approach outperforms up-to-date methods for the same problem, having a real-time prediction performance at 45 fps and 150 fps (fast-yolo), reasoning globally about the image, encoding information about classes and appearance. However, YOLO has more localization errors like small objects, but it is less likely to predict false positives.

2.2 Unified Detection and Network Architecture

Yolo introduces the concept of unified detection, which consist of unifying object detection and bounding box regression into a single neural network. This network divides the input image in a grid of SxS cells, and per cell, the model predicts the bounding box (x, y, w, h) and a confidence score. This approach offers several advantages, including faster inference time and the ability to learn robust representations of objects by considering them in context. Moreover, it predicts a conditional class probability that quantifies the probability that the object in the cell belongs to a specific class. Therefore, the confidence score is computed using the following equation [1].

$$Pr(cls_i|obj) * Pr(obj) * IoU^{pred}_{truth} = Pr(cls_i) * IoU^{pred}_{truth} \tag{1}$$

The YOLO network architecture is based on GoogLeNet for image classification, consisting of 24 convolutional layers, and a max-pooling layer in charge of extracting features followed by two fully connected layers to predict output probabilities and bounding box coordinates. The authors introduce the architecture of Fast YOLO with only nine convolutional layers and fewer filters to achieve 150 fps processing [1].

2.3 Experimental Evaluation and Training

YOLO network with only 20 conv-layers, an average pooling layer, and one fully connected layer is pre-trained on ImageNet 1000 class classification for a week, achieving a top-5 error of 88%. Afterward, the model was modified for object detection adding four conv-layers and two fully connected layers and finally predicting the class probabilities and bounding boxes. The optimizer used was a sum-squared error for easy optimization and improved the instability of the model with two parameters that increase the loss of bounding boxes and reduce the loss for confidence predictions to have an optimizable combination of classification loss and localization loss. Data augmentation was used to improve model accuracies like random sampling, translations, and adjusting image exposure and saturation. Another technique to avoid overfitting is dropout after the first fully connected layer to prevent co-adaptation between layers. The training of the network was done in 135 epochs using the PASCAL VOC 2007/2012 with a batch size of 64, a momentum of 0.9 aiming to converge smoothly and faster; weight decay of 0.0005, and the authors also used a schedule learning rate to reduce the learning rate in the process [1].

In terms of performance, YOLO is fast because requires one single network evaluation to predict object location, the grid design leads to spatial diversity, and the authors used non-max suppression to fix multiple detections in large objects. However, YOLO has limitations in detecting small objects, crowded objects, and model instability. YOLO outperformed other methods of object detection like Fast-RCNN and RCNN, achieving a mean average precision (mAP) of 63.4% on the PASCAL VOC 2012 in 45 frames per second (FPS) on Titan X GPU, and combining YOLO with Fast-R-CNN achieves 75.0%. YOLO achieved a real-time detection system with competitive accuracy [1].

2.4 Conclusion

In conclusion, YOLO is an end-to-end method for object detection that unifies object classification and bounding box regression into a single neural network, resulting in faster inference times and robust representations of objects [1]. The YOLO network architecture is based on GoogLeNet, with modifications to predict both class probabilities and bounding boxes. The model was pre-trained on ImageNet and fine-tuned on PASCAL VOC datasets using data augmentation and dropout techniques to improve model accuracy and avoid overfitting. Despite its limitations in detecting small and crowded objects, YOLO outperforms other object detection methods in terms of speed and achieved a real-time detection system with competitive accuracy [1]

3 DeepLabv3

This paper addresses two challenges of image segmentation, the reduced feature map resolution, and segmentation at multiple scales. They introduce Atrous convolutions to enlarge the field of view of feature maps without adding more parameters, additionally introduces the Atrous Pyramid Pooling (ASPP) Module to capture features at multiple scales dividing the feature maps into regions to apply pooling operations [2]. There are different types of fully convolutional networks to capture multi-scale context like Image Pyramid and encoder-decoder aim the capture of long-range context and object details. Also, Spatial Pyramid Pooling aggregates global context by capturing context at several ranges [2].

Deep Convolutional Neural Networks are effective in performing image segmentation. However, methods like stride and pooling operations reduce the spatial resolution of feature maps affecting the performance of the application. Deconvolutional layers address the problem but add more parameters to the architecture. One of the methods that this study introduces is the Atrous Convolutions to control the density of feature responses and adjust the receptive field in convolutional neural networks without adding extra parameters [2].

3.1 Atrous Convolutions and Atrous Spatial Pyramid Pooling

Atrous convolution redefines the convolution applied over an input x with a filter, it increases the filter's receptive field by inserting zeros between consecutive filter values. The amount of holes inserted is controlled by the Atrous rate allowing modification of the field of view of the filters. The authors also use this definition to control the spatial dimension of feature responses, spatial density leading the capture of multi-scale contextual information. Normally, the output stride of a traditional CNN is 32 presenting a problem for segmentation task because it requires feature maps with a higher spatial resolution to accurately segment objects. The paper uses Atrous convolutions to control the spatial density of feature responses, applying different atrous rates in different layers to control the effective output stride, an output stride of 16 will double the spatial density of a standard convolutional neural network. This allows the extraction of dense feature responses without additional parameters or reducing the spatial resolution of output maps [2].

One of the first implementations of the paper of Atrous convolutions is as a cascade structure using ResNet blocks up to block 7, each of them consist 3 convolutional operations of 3x3 per block with a stride of 2 at the end. Within blocks 4 to 7 it is applied the Atrous convolutions in a multi-grid structure (r_1, r_2, r_3) adopt different dilatation rates, controlling the output stride. The final Atrous rates are calculated with the multiplication of the unit rate and the corresponding rate. Another proposed method by the authors is the Atrous Spatial Pyramid Pooling consisting of 4 parallel Atrous convolutions on top of the feature map with different Atrous rates and includes batch normalization. The ASPP proposed is defined as a 1x1 convolution, 3 Atrous convolutions with 3x3 filter size, 256 filters, and (6,12,18) rates for an output stride of 16, followed by concatenation with another 1x1 convolution [2].

3.2 Experimental Evaluation and Training

The DeepLabv3 paper provides a detailed account of the experimental evaluation and training of the model. The authors evaluate their model on two datasets, PASCAL VOC 2012 and Cityscapes, using a standard evaluation protocol. They use a trainval dataset for training and evaluate on a test dataset. The evaluation metric used in both datasets is mean Intersection over Union (mIoU) to measure the overlap between the predicted and ground truth segmentation [2].

The authors use data augmentation techniques such as random scaling (0.5 to 2.0) and random left-right flipping. They use a batch size of 16 and weight decay of 0.997. The Atrous convolutions allow control of the output stride during training. Initially, an output stride of 16 is used for 30k iterations to allow faster convergence. In the second part of the training, an output stride of 8 is used, and the learning rate is reduced to 0.001 using the PASCAL trainval set [2].

In the performed experiments, the authors use ResNet-50 adding block 5, 6, and 7 within a cascade structure. The performance tends to improve by adding more cascade blocks. ResNet-101 is also used with cascade blocks, but the performance has a small margin of difference. However, going deeper with multi-grid improves the performance [2].

The authors test the other method using ASPP in block 4, which has a better performance of 79.35%, making it the final model for test evaluation. The addition of other parallel branches evidenced a reduction in performance. The inference strategy evidenced that the output stride of 8 allows a detailed feature map, improving the performance from the output stride of 16. After training, the authors adopt multi-scale inputs and left-right flipped images to improve the final performance [2].

3.3 Conclusion

The DeepLabv3 paper concludes that the cascade and ASPP outperform DeepLabv2 by providing a better way to encode multi-scale context[2]. Additionally, using the ASPP model on MS-COCO achieved a performance of 82.7% on the validation set with an output stride equal to 8 and the same configuration parameters. However, there exist difficulties in the model for segmenting some objects. Furthermore, the pre-trained model using ImageNet and JFT-300M dataset resulted in a performance of 86.9% on the PASCAL VOC 2012 test [2].

The DeepLabv3 paper proposes a highly effective architecture for semantic segmentation. By using Atrous convolutions, the authors address the issue of reduced feature resolution and the challenge of segmenting objects at multiple scales. The Atrous Pyramid Pooling Module allows the network to capture features at multiple scales by dividing the feature maps into regions and performing pooling operations. The experimental results indicate that the proposed architecture outperforms existing state-of-the-art methods for semantic segmentation.

4 Comparison of Segmentation and Object Detection Tasks and Techniques

Segmentation and object detection are two related but different tasks in computer vision.YOLO is one of the pioneers in object detection which identify and locate objects in an image by providing bounding box coordinates that enclose the object. On the other hand, DeepLabv3 offers a method for image segmentation that create parts or segments from the input image that represents the complete or part of the object, the segmentation task assigns a class label to a pixel value. Then, the two tasks are related to detecting the objects in an image they differ in the implementation, while object detection performs a regression to predict bounding boxes, the segmentation task uses the regression to perform pixel-level classification.

In terms of differences, object detection divides the image to perform feature extraction and object proposal generation, followed by classification with regression to obtain bounding boxes. YOLO performs this detection in a single track using a single convolutional network to predict the object class probabilities and bounding box coordinates, the main benefit of this method is the real-time performance of the model, up to 145 fps while maintaining a competitive accuracy. YOLO uses a unified detection system that creates region proposals responsible for detecting the object and the coordinates of the box that contains the object. On the other hand, DeepLabv3 uses fully convolutional layers with Atrous convolutions and Atrous pyramid pooling layers to predict pixel-wise class with an output of a pixel-segmentation mask. This type of convolution increases the receptive field and captures context information without the need to increase parameters, additionally, this type of convolution allows to control of the spatial density that leads to segmenting small objects keeping small details. Both studies globally use overfitting techniques, data augmentation, and pre-training to improve performance. However, in specific, they differ in the implementation, configuration, and parameters.

References

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[2] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation. arxiv 2017," *arXiv preprint arXiv:1706.05587*, vol. 2, 2019.