YOLO를 이용한 이미지 Blur 처리

강동연
한양대학교 컴퓨터 공학과
e-mail:dongykang@gmail.com

Blur the objects in image by YOLO

Dongyeon Kang
Dept of Computer Science, Han-yang University

Abstract

In the case of blur processing, it is common to use a tool such as Photoshop to perform processing manually. However, it can be considered very efficient if the blur is processed at one time in the object detection process. Based on this point, we can use the object detection model to blur the objects during the process. The object detection is performed by using the YOLO [3] model. If such blur processing is used, it may be additionally applied to streaming data of video or image.

1. Introduction

Object detection algorithms are improved since CNN [1]. In terms of accuracy and speed, many algorithms developed using deep learning. The goal of object detection is to determine whether or not there are any instances of the given categories (such as humans, cars, bicycles, dogs and cats) in some given image. Object detection using Deep Learning can be divided into two major directions, 1-stage object detector which based on regression/classification and 2-stage object detector based on region proposal. There are popular object detect algorithms named Faster R-CNN [2], YOLO [3], SSD [4 etc]. We will use YOLO among object detection models for blur processing. Considering the performance of YOLO, the latest version, we used YOLO-v3 [6] which has good performance in terms of speed and accuracy.

We will look at representative three models. First, after reviewing SSD [4], Faster R-CNN [2], we will explore YOLO [3] in more detail. And then, We will focus on the performance difference between YOLO-v3 [6] and existing YOLO, and discuss experimental results.

2. Related works

Region proposals detected with the Selective Search method were still necessary in the previous model [7], [8], which is computationally expensive.

Faster R-CNN have introduced Region Proposal Network (RPN) to directly generate region proposals [2], predict bounding boxes and detect objects. The Faster Region-Based Convolutional Network (Faster R-CNN) is a combination between the RPN and the Fast R-CNN model. Overview of Faster R-CNN is in Figure 1.

1) Faster R-CNN: CNN model takes the entire image as input and generates a feature map. A 3 x 3 sized window slides through all feature maps and outputs a feature vector linked to two fully regressed layers one for box regression and one for box classification. The output of the box regression layer is 4k in size (box's coordinates, height and width) and the output size of the boxed classification layer is 2k (detecting objects or objects in the "objectness" score box) [2]. The k region proposal detected in the sliding window is called an anchor. Figure 2 shows detecting the anchor boxes for a single 3 x 3 window. When the anchor boxes are detected, they are selected by applying “objectness” score to keep only the relevant boxes. Faster R-CNN uses RPN [2] to avoid the Selective Search [5] method, it accelerates the training and testing processes, and improve the performances. The architecture of RPN is shown in Figure 2.

2) SSD (Single Shot Multibox Detecotr): It presents an object detection model using a single deep neural network combining regional proposals and feature extraction. SSD to predict all at once the bounding

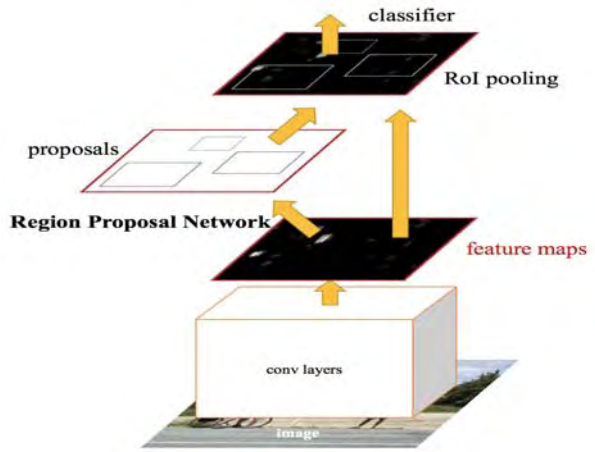

Figure 1. Overview of Faster R-CNN [6].

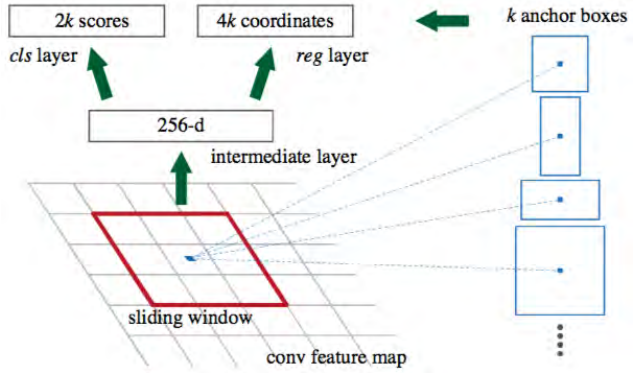

Figure 2. The RPN in Faster R-CNN [6]. K predefined anchor boxes are convoluted with each sliding window to produce fixed-length vectors which are taken by classifications and regression layer to obtain corresponding outputs.

boxes and the class probabilities with a end-to-end CNN architecture. The model takes an image as input which passes through multiple convolutional layers with different sizes of filter [4]. Feature maps from convolutional layers at different position of the network are used to predict the bounding boxes. They are processed by a specific convolutional layers with 3 x 3 filters called extra feature layers to produce a set of bounding boxes similar to the anchor boxes of the Fast R-CNN [8]. Each box has 4 parameters: the coordinates of the center, the width and the height. At the same time, it produces a vector of probabilities corresponding to the confidence over each class of object. The Non-Maximum Suppression method [3] is also used at the end of the SSD model to keep the most relevant bounding boxes. The Hard Negative Mining [4] is then used because a lot of negative boxes are still predicted. It consists in selecting only a subpart of these boxes during the training. The boxes are ordered by confidence and the top is selected depending on the ratio between the negative and the positive which is at most 1/3.

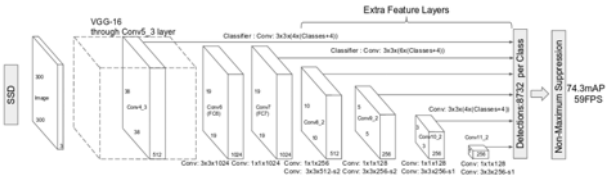

Figure 3. The SSD network leveraging feature maps from VGG-16 [4].

3) YOLO (You Only Look Once)

The YOLO is novel approach for end to end object detection: instead of re-purposing classification as a way to address object detection, YOLO frames detection as a regression problem with spatially separated bounding boxes and associated class probabilities. Using this formulation, a single neural network can be used to predict the bounding boxes and class probabilities directly from the full image in one step evaluation [3]. Since the entire detection pipeline is a single network, it can consequently be optimized end-to-end directly on detection performance.

The basic architecture is using a resize step, a series of convolution steps and non-max suppression. The network predicts multiple bounding boxes in one run, as well as the class probabilities for those boxes. The overall architecture is presented in Figure 4.

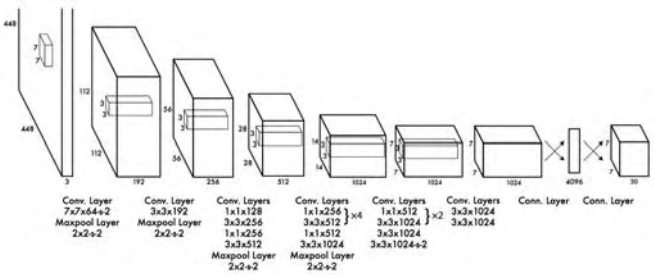

Figure 4. The architecture of YOLO [2].

Each bounding box contains 5 elements: (x, y, w, h) and a box confidence score. The confidence score reflects how likely the box contains an object and how accurate is the bounding box. We normalize the bounding box width w and height h by the image width and height. The x and y are offsets to the corresponding cell. Hence, x, y, w and h are all between 0 and 1. Each cell has 20 conditional class probabilities. The conditional class probability is the

probability that the detected object belongs to a particular class. The class confidence score for each prediction box is computed as:

class confidence score = box confidence score x conditional class probability

It measures the confidence on both the classification and the localization. The loss in YOLO adds three different components (localization, confidence and classification losses together).

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$
$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

Figure 5. Loss of YOLO [2].

Classification loss at each cell is the squared error of the class conditional probabilities for each class. The localization loss measures the errors in the predicted boundary box locations and sizes. We only count the box responsible for detecting the object.

YOLO can make duplicate detections for the same object. To fix this, YOLO applies non-maximal suppression [3] to remove duplications with lower confidence. 1) Sort the predictions by the confidence scores. 2) Start from the top scores, ignore any current prediction if we find any previous predictions that have the same class and IoU>0.5 with the current prediction. 3) Repeat step 2 until all predictions are checked.

3. Result

We use the YOLO-v3 model for training and pre-trained weights for training. YOLO-v3 shows the latest performance when compared with the performance of the representative models of object detection mentioned above [2], [3], [4], [6]. The most salient feature of YOLO-v3 is that it makes detections at three different scales. The detection is done by applying 1 x 1 detection kernels on feature maps of three different sizes at three different places in the network. The shape of the detection kernel is 1 x 1 x (B x (5 + C)). Where B is the number of bounding boxes that the cell can predict in the feature map, "5" is for the 4 bounding box attributes and one object confidence, and C is the number of classes. In YOLO-v3 trained on COCO [9], B = 3 and C = 80, so the kernel size is 1 x 1 x 255. The feature map produced by this kernel has identical height and width of the previous feature map, and has detection attributes along the depth as described above.

Detections at different layers helps address the issue of detecting small objects, a frequent complaint with YOLO-v2 [5]. The up-sampled layers concatenated with the previous layers help preserve the fine grained features which help in detecting small objects [10]. YOLO-v3, in total uses 9 anchor boxes. Arrange the anchors is descending order of a dimension. Assign the three biggest anchors for the first scale , the next three for the second scale, and the last three for the third.

For an input image of same size, YOLO-v3 predicts more bounding boxes than YOLO-v2. YOLO-v3 predicts 10x the number of boxes predicted by YOLO-v2 [5], [6]. Also, there is a change in loss function. In YOLO-v3, localization, confidence and classification losses have been replaced by cross-entropy error terms.

YOLO-v3 now performs multi-label classification for objects detected in images. In YOLO, authors used to softmax the class scores and take the class with maximum score to be the class of the object contained in the bounding box [3]. If an object belongs to one class, then it cannot belong to the other. This works fine in COCO dataset. Each class score is predicted using logistic regression and a threshold is used to predict multiple labels for an object. Classes with scores higher than this threshold are assigned to the box [6].

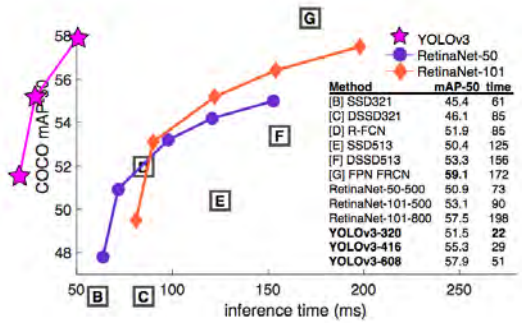

Figure 6. Performance on COCO 50 Benchmark.

YOLO-v3 works equally well with other state of art detectors in Figure 6. It is also better than SSD and it's variants.

After detecting object in image with YOLO-v3, it processed blur on the bounding box by simple trick. When drawing bounding boxes and labels, we add blur by using the gaussian blur in Opencv. There's no other additive time or space for blur the objects.

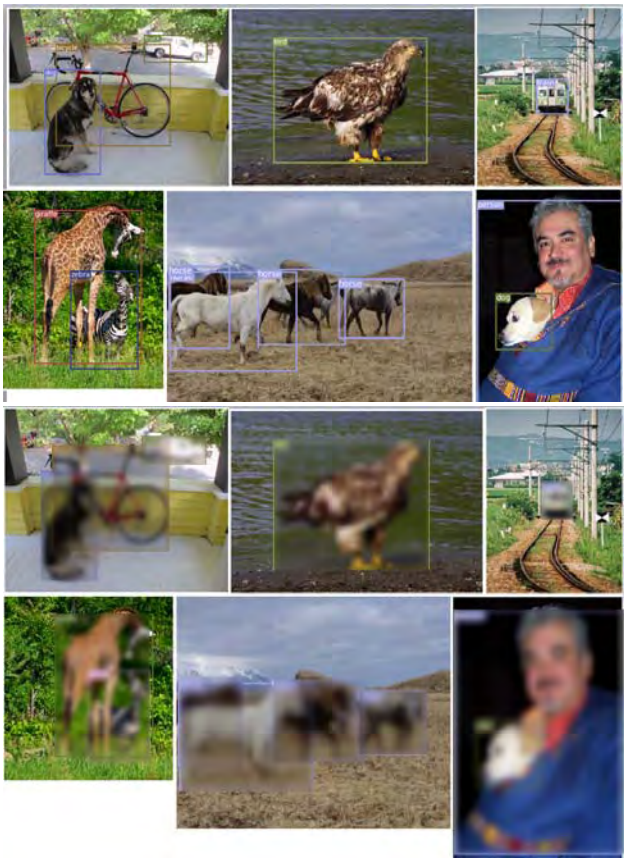

Figure 8. Result of blur processing in image with object detecting.

Figure 8 images are the result of YOLO-v3 with pretrained weights. We can make multiple blurs on image by detection. However, the problem is that when a class is overlapped, it cannot know which class has been executed when it is blurred. Futhermore, we can apply this methods to video. If it improves gradually, it will be much more effective and efficient than mosaic or blur each image in real time streaming video or on a platform such as YouTube.

4. Conclusion

For blur processing through object detection, representative object detection models are discussed. In this paper mainly looked at the YOLO model in detail. This paper first explores and experiments with emphasis on performing blur processing on objects per image. In the future, we can make real time blur processing by using this method. Based on this method, If we improve the additional details for blur processing by the desired class in the image, it is also possible to perform real-time image processing on-line.

참고문헌

[1] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, Matti Pietikainen, "Deep Learning for Generic Object Detection: A Survey" arXiv:1809.02165, 2018.

[2] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks" in NIPS, 2015.

[3] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection" in CVPR, 2016.

[4] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single shot Multibox Detector" in ECCV, 2016.

[5] J. Redmon and A. Farhadi. "YOLO9000: Better, Faster, Stronger" in CVPR, 2017

[6] J. Redmon and A. Farhadi. "YOLOv3: An Incremental Improvement" arXiv:1804.02767, 2018.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich feature hierarchies for accurate object detection and semantic segmentation" in CVPR, 2014.

[8] R. Girshick, "Fast R-CNN" in ICCV, 2015.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Doll´ar, and C. L. Zitnick, "Microsoft COCO: Common Objects in Context" in ECCV, 2014.

[10] J. Long, E. Shelhamer, and T. Darrell, "Fully Convolutional Networks for Semantic Segmentation" in Proc. Comput. Vis. Pattern Recognit., 2015.