Master's Thesis of Science

Efficient CNN-based detection of occluded and nearby objects

겹친 물체와 근거리 물체 검출을 위한 효율적인 CNN기반 방법

February 2019

Graduate School of Engineering
Seoul National University
Electrical and Computer Engineering Major

Yin Xiaopeng

Efficient CNN-based detection of occluded and nearby objects

Supervisor: Prof. Hyuk-Jae Lee

Submitted as master's thesis of Science

February 2019

Graduate School of Engineering
Seoul National University
Electrical and Computer Engineering Major

Yin Xiaopeng

Confirming the master's thesis written by
Yin Xiaopeng
February 2019

Chair	Nam-Ik Cho	(Seal)
Vice Chair	Hyuk-Jae Lee	(Seal)
Examiner	Jong-Han Lim	(Seal)

Abstract

Object detection, which primarily recognizes the category, location, and the number of objects in an image, is an important research field of computer vision. Recently, the state-of-art object detection network mainly take advantage of Convolutional Neural Network(CNN) algorithm and generally get better performance compared to traditional non-CNN-based object detection algorithms. However, when objects in image are overlapped with or close to other objects, the CNN-based network cannot generate accurate bounding boxes, and the network sometimes even generates extra wrong or useless bounding boxes that leads to wrongly counting how many objects in image. This work proposes a post processing algorithm for improving the detection accuracy and eliminating the wrong and redundant bounding boxes. The post processing algorithm consists of three steps: First, rotating the image with arbitrary degree and inputting the rotated images and original images into CNN network to generate enough bounding boxes from original and rotated images. Then, mapping all bounding boxes in rotated images back to original image with proposed bounding box mapping algorithm. However, with the increase of rotated images, the inference time increase as well. Hence, this work proposed a separated YOLO, which uses a tiny YOLO to determine the ROI, then uses another smaller YOLO to do the

bounding boxes prediction. By this way, the inference time is decreased significantly. Finally, using proposed non-maximum suppression(NMS) algorithm to eliminate the redundant bounding boxes. With post-processing algorithm, the detection accuracy increase significantly.

Keyword : object detection, non-maximum suppression(NMS), you look only once(YOLO).
Student Number : 2017-20513

Table of Contents

List of Figures

List of Tables

Chapter 1. Introduction

1.1. Object Detection Review

Computer vision research primarily consists of four core problems. Image classification: given an image, point out what are objects in the image. Localization: localize the objects in image. Detection: the research combines localization and classification, in other words, given an image, the detection algorithm generates the position of object and its category. Segmentation: the segmentation research includes instance segmentation and scene segmentation, so given an image, the segmentation algorithm need to point out to which object or scene every pixel of image may belongs.

The main task of object detection is find out all interested objects in image, and determine their position, size and category. Due to different shape, appearance and posture of object, illumination and occlusion, the object detection is always a challenge in computer vision field.

Generally, when we human beings look at an image, we can instantly tell what kinds of object are, and where these objects are in this image. The visual system of human is fast and accurate so that we expect that we can develop a fast and accurate object detection algorithm to simulate human's visual system. Hence, the algorithm for

object detection should be fast and accurate, allowing people to implement it in daily life with a real-time speed, which is normally defined as greater than 30 frames per second (FPS). However, there is always a trade-off between speed and accuracy. So finding the balance between speed and accuracy is important for object detection algorithm and network development, which is also the popular research in computer vision the majority of researchers are concentrating on.

Nowadays, the object detection algorithm is divided into two different groups. One group is based on traditional image processing algorithm and machine learning. This group includes many classic detection algorithm, such as Haar-like and Adaboost algorithm [1, 2, 3], HoG and SVM algorithm [4, 5, 6, 7], DPM algorithm [8, 9, 10, 11, 12]. All these methods are using sliding window to acquire enough bounding boxes, and using image pyramid to generate multiple bounding box for same position, then to select the best bounding boxes with highest score or confidence and most suitable size, which means the bounding box we finally select could surround the object accurately. But these traditional methods are slow and not accurate enough because of numerous sub-images produced by sliding window and the huge computation, increasing people are abandoning them and turning to another group motivated by deep learning.

Deep learning comprises convolutional neural network (CNN), recurrent neural network (RNN) and reinforcement learning. CNN is one type of feedforward neural network whose artificial neurons could

respond to a part of covered surrounding units and perform well on large image processing. CNN has three main layers, namely convolutional layer, pooling layer and activate layer. Besides, the basic structure of the CNN includes two layers, one of which is a feature extraction layer that includes convolutional layer and pooling layer, and the input of each neuron is connected to the local receptive filed of the previous layer, and the local features are extracted. Once the local features are extracted, its positional relationship with other features is also determined; the second is the feature mapping layer, also called activate layer, each computing layer of the network is composed of multiple feature maps, and each feature map is a plane. All neurons in the plane have equal weights. The feature mapping structure uses a small sigmoid function that affects the function kernel as the activation function of the convolutional network, so that the feature map has displacement invariance. In addition, since the neurons on one mapping surface share weights, the number of network free parameters is reduced. Each convolutional layer in the convolutional neural network is followed by a computational layer for local averaging and quadratic extraction. This unique two-feature extraction structure reduces feature resolution.

CNN is mainly used to identify two-dimensional images of displacement, scaling and other forms of distortion invariance. This part of the function is mainly realized by the pooling layer. Since CNN's feature detection layer learns through training data, when CNN is used,

explicit feature extraction is avoided, and learning is implicitly learned from training data. Furthermore, due to neuron weights on the same feature mapping surface are same, so the network can learn in parallel, which is also a major advantage of convolutional networks relative to the network of neurons connected to each other. Convolutional neural networks have unique advantages in image processing and computer vision due to their special structure of local weight sharing. Their layout is closer to the actual biological neural network, and weight sharing reduces the complexity of the network, especially multidimensional. The feature that the input vector image can be directly input into the network avoids the complexity of data reconstruction during feature extraction and classification.

As the development of CNN, the performance of object detection has got a significant improvement. The Second group is, hence, mainly based on deep learning logically, which is also divided into two groups, namely one-stage network and two-stage network. One-stage network is also called as end-to-end network, this group has YOLO [13, 14, 15], SSD [16] that both are famous for the inference speed. Two-stage network, which known as region proposal network, has higher detection accuracy and is much slower than one-stage network. R-CNN [17, 18, 19] and SPP-NET [20] stand for this two-stage network.

Two-stage networks like R-CNN use a sliding window like method – region proposal methods as first stage to generate a mass of potential bounding boxes, and then run the second stage – CNN-based classifier

on the boxes produced in first stage to determine the category of each box and to tune the coordinates of each box making the bounding box more accurate. However, for this two-stage network, we need feed every box of proposal boxes generated by region proposal net into the second stage both at training process and inference process, so these two-stage networks are too computationally intensive and too slow compared with one-stage network while more accurate.

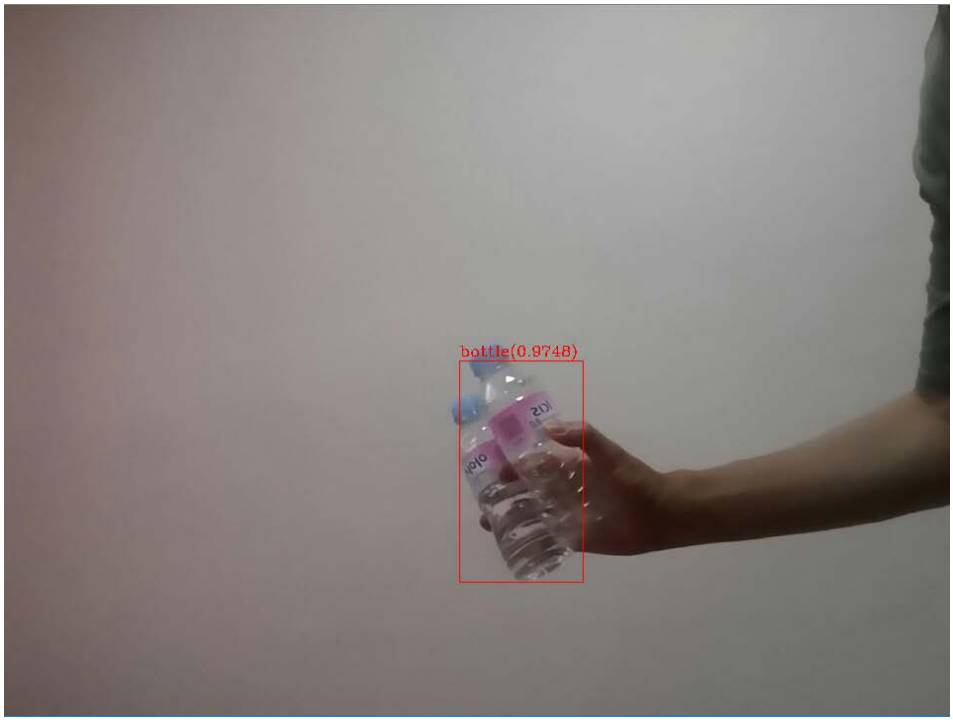
To detect in real-time, people would more like to adopt one-stage network and YOLO with speed 45 FPS is usually the first choice.

1.2. Purpose of Research

While high speed and real-time, YOLO sacrifices its detection accuracy. Especially when objects are close to each other or objects are so close that some objects are overlapped by others. Under such situation, YOLO has a comparably disappointing performance.

In general, when objects in image are close or overlapping, YOLO may generate bounding boxes that one extra box encloses two objects, and this extra box will be remained and the correct boxes will be eliminated after post-processing, shown as Figure 1-1.

a)



b)

Figure 1-1. a) detection result of YOLO b) result after post-processing

On the other hand, YOLO sometimes cannot detect all interested objects in image when objects are close or overlapping, shown as Figure 1-2.

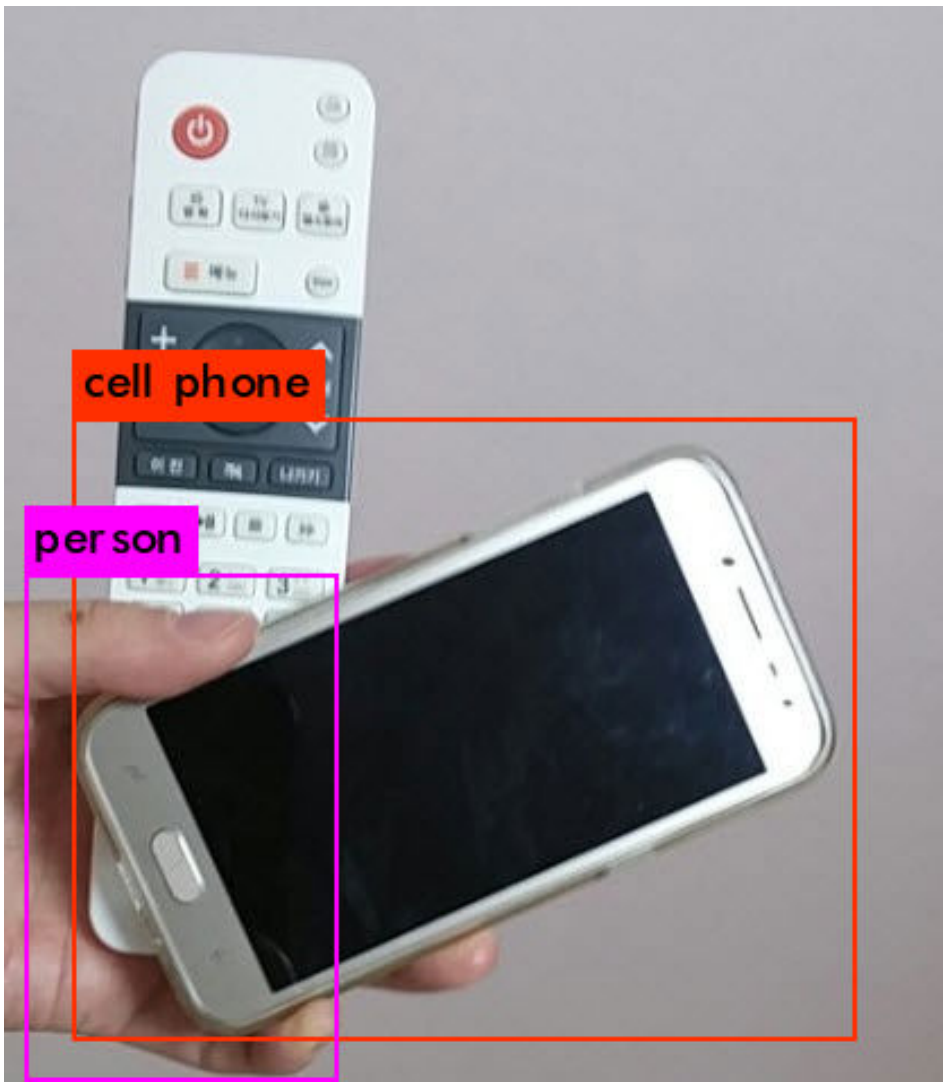


Figure 1-2. The result of YOLO with one missed object

This work is to improve detection accuracy and solve the detection problems when objects in image are close and overlapping. To this end,

this work proposes a new detection method and novel post-processing algorithm.

Chapter 2. YOLO and Post-Processing Algorithm

2.1. YOLO Detection Network

Compared to two-stage detection network, like Faster R-CNN, YOLO provides another different idea that transfers object detection problem to regression problem. Given the input image, the bounding box of the target and its classification category are directly regressed at multiple locations of the image. YOLO is a convolutional neural network that predicts multiple bounding boxes' locations and categories at once. It enables end-to-end object detection and recognition. Its biggest advantage is its speed. In fact, the essence of object detection is regression, so a CNN that implements regression does not require complicated design processes. YOLO does not adopt the sliding window or the method of extracting the proposal region to train the network, but directly selects the whole picture training model. The advantage of this method is that it can better distinguish between the object and the background area. In contrast, Faster R-CNN, which takes region proposal as first stage, often mis-detects the background area as a specific object. The network structure of YOLOv1 [13] are shown as Figure 2-1.

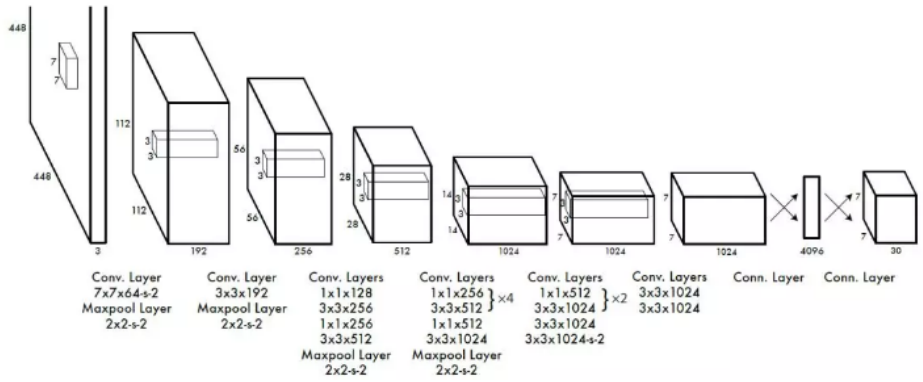


Figure 2-1. YOLO structure

2.1.1. YOLO Version 1

YOLO divides the input image into S*S grids, each of which is responsible for detecting objects that the center of object falls in the grid. Each grid predicts B bounding boxes and the confidence scores

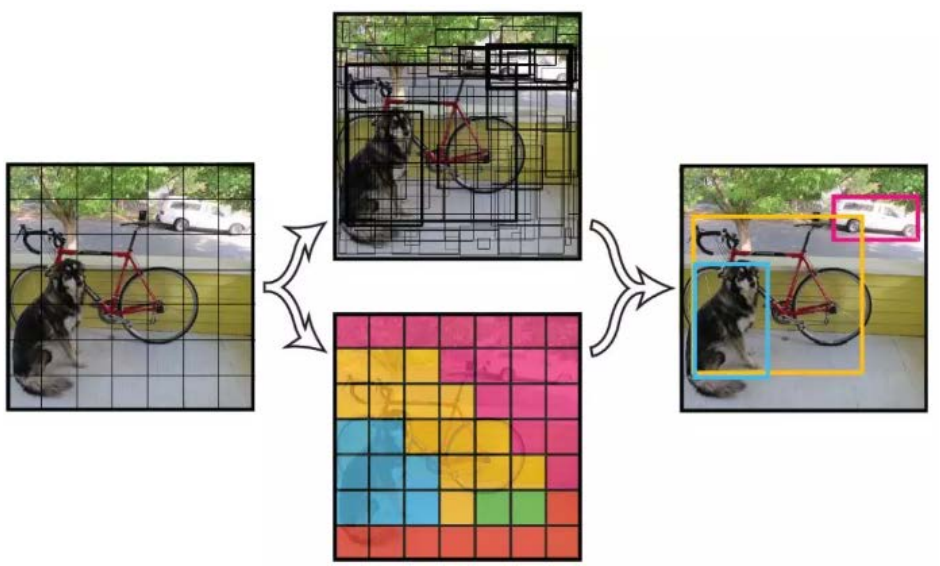


Figure 2-2. YOLO model

for those bounding boxes. This confidence scores reflect the model's

$$\mathbf{c} = \mathbf{Pr}(\mathbf{obj}) * \boldsymbol{IoU}(\boldsymbol{pred}, \boldsymbol{truth})$$

predictions for this grid: whether the grid contains objects and how accurate the coordinates of the box are. The model of YOLO shown as Figure 2-2.

For each bounding box, YOLO predicts 5 values (x, y, w, h) and confidence, and for each grid, YOLO also predicts a category information, which is denoted as class C. Then SxS grids, each grid predicts B bounding boxes and predicts C categories. The output is a tensor of SxSx(5*B+C).

At test stage, the class of each grid’s prediction is multiplied by the confidence predicted by the bounding box to obtain the class-specific confidence score of each bounding box:

$$\begin{aligned} &\Pr(\text{class}|\text{obj}) * \Pr(obj) * IoU(pred, truth) \\ &= \Pr(class) * IoU(pred, truth) \end{aligned}$$

At training stage, the loss function of YOLO is:

$$\lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathrm{I}_{i,j}^{obj}[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2]$$

$$+\ \lambda_{coord}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathrm{I}_{i,j}^{obj}\left[\left(\sqrt{x}_i-\sqrt{\hat{x}_i}\right)^2+\left(\sqrt{y}_i-\sqrt{\hat{y}_i}\right)^2\right]$$

$$+\ \sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathrm{I}_{i,j}^{obj}\left(C_i-\hat{C}_i\right)^2$$

$$+\ \lambda_{boobj}\sum_{i=0}^{s^2}\sum_{j=0}^{B}\mathrm{I}_{i,j}^{obj}\left(C_i-\hat{C}_i\right)^2$$

$$+\ \sum_{i=0}^{S^2}\mathrm{I}_{i}^{obj}\sum_{c\in classes}(p_i(c)-\hat{p}_i(c))^2$$

For loss function, the classification error is only penalized when there is an object in the grid and only when a bounding box predictor is responsible for a ground truth bounding box will be penalized for the box's coordinate error, and which ground truth box the bounding box will be responsible for is based on its predicted value and whether the IoU between predicted bounding box and ground truth bounding box is the largest for all predicted bounding boxes in this cell or grid.

2.1.2. YOLO Version 2

YOLO version 1 has many shortcomings, for example, when objects are close or small, the performance of YOLO is not good enough. Hence the author hopes that the direction of improvement is to improve recall and improve the accuracy of localization while

maintaining the accuracy of classification.

First, YOLO v2 optimizes the network using Batch Normalization to increases the convergence ability of the network while eliminating the need for other forms of regularization. By adding Batch Normalization to each of YOLO's convolutional layers, it eventually increases mAP by 2% while also regularizing the model. Using Batch Normalization, we can remove Dropout from the model without overfitting.

Then, in order to adapt to the new resolution, YOLO v2's classification network first fine-tunes on ImageNet with a resolution of 448*448, roughly 10 epochs, which allows the network to adjust the filters so that it can run better. For the new resolution, you also need to tune the Resulting Network for detection. Finally, by using high resolution, mAP has increased by 4%.

The YOLO v1 uses fully connected layers that directly predict the coordinates of Bounding Boxes. However, the Faster R-CNN method uses only the convolutional layer and the Region Proposal Network to predict the Anchor Box offset and confidence, rather than directly predicting the coordinates. The authors found that by predicting the offset rather than the coordinate values, the problem can be simplified and the neural network can be learned more easily.

So, YOLO v2 removes the fully connected layer and adopts Anchor Boxes to predict Bounding Boxes. The author removes some pooling layers in the network, which allowed the output of the convolutional layer to have a higher resolution. Since the objects in the picture tend

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

Figure 2-3. The model structure of YOLO v2

to appear in the center of the picture, especially the larger objects, there is a specific location at the center of the object to predict these objects. YOLO's convolutional layer uses a value of 32 to downsample the picture, so by selecting 416*416 as the input size, it can finally output a

13*13 feature map which is 8*8 in YOLO v1. Using the Anchor Boxes will make the accuracy slightly lower, but using it will allow YOLO to predict more than a thousand boxes, with a recall of 88% and a mAP of 69.2%.

YOLO v2 is based on a new classification model - Darknet19, which is somewhat similar to VGG. YOLO v2 uses a 3*3 filter that doubles the number of Channels after each pooling layer. YOLO v2 uses global average pooling, and batch normalization to make training more stable, accelerate convergence, and normalize the model. Figure 2-3 shows the model structure of YOLO v2.

2.1.3 YOLO Version 3

YOLO v3, based on Darknet53 which much deeper than Darknet19 used in YOLO v2, is much more complicated than the previous model, and the speed and accuracy can be balanced by changing the size of the model structure.

YOLO v3's Prior Detection system reuses the classifier or locator for inspection tasks. They apply the model to multiple locations and scales of the image. Those areas with higher scores can be considered as detection results. In addition, YOLO v3 uses a completely different approach compared with other target detection methods. YOLO v3 applies a single neural network to the entire image, which divides the image into different regions, then predicts the bounding boxes and probabilities for each region, and the predicted bounding boxes are

weighted by the predicted probabilities. YOLO v3 has some advantages over classifier-based systems. It looks at the entire image as image is tested, so its predictions take advantage of the global information in the image. Unlike R-CNN, which requires thousands of single-target images, YOLO v3 predicts through a single network assessment. This makes YOLO v3 very fast, generally 1000 times faster than R-CNN and 100 times faster than Fast R-CNN.

YOLO v3 predicts bounding boxes at 3 different scales, and the anchor design is still clustered, and nine cluster centers are finally obtained, which are equally divided into three scales according to size.

- Scale 1: Add some convolutional layers after the underlying network and output the box information.

- Scale 2: Sampling from the convolutional layer of the penultimate layer in scale 1 (x2) and then adding the feature map of the last 16x16 size, and outputting the box information again after multiple convolutions, which is larger than the scale 1 double.

- Scale 3: Similar to Scale 2, using a 32x32 feature map.

The model structure of YOLO v3 is shown in Figure 2-4.

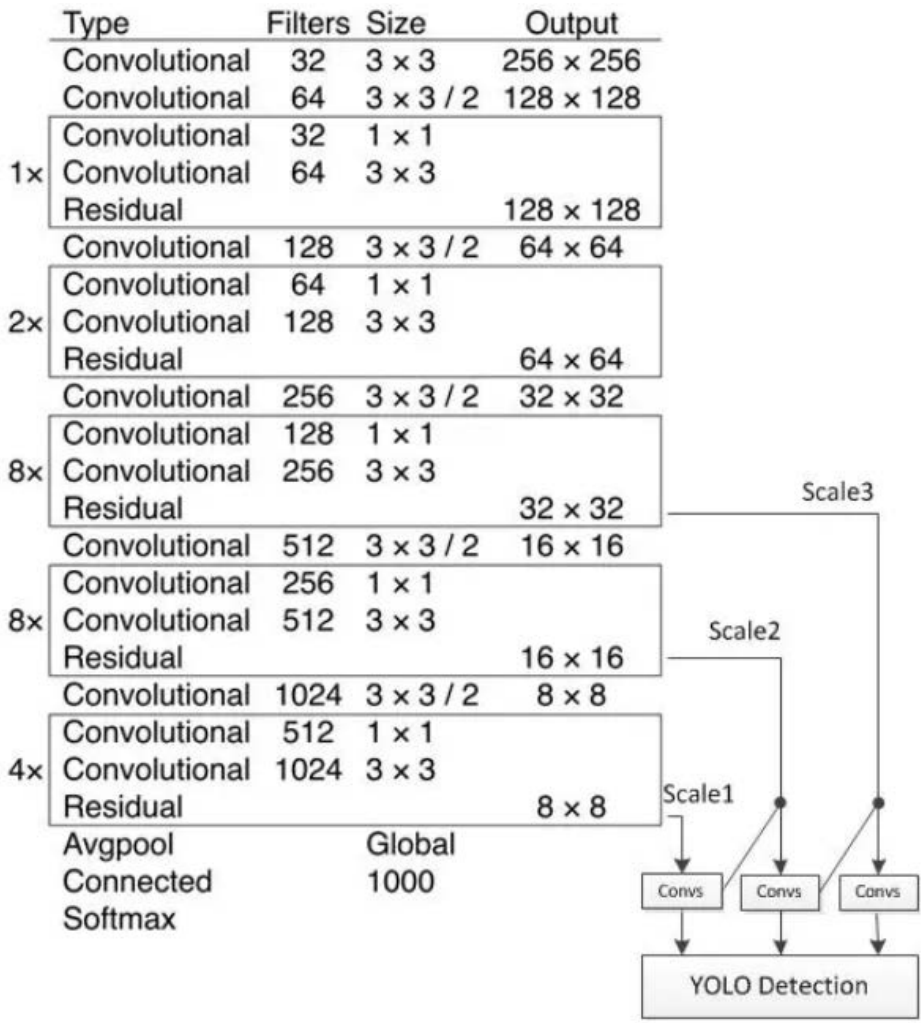


Figure 2-4. The structure of YOLO v3

while YOLO is fast and comparably good precision and recall on object detection task, it still has some defects, especially for small objects, occluded objects and nearby objects.

For example, when objects are very close, YOLO always recognize these nearby objects as one object. On the other hand, when objects are small or occluded, YOLO has difficulties on localizing these objects, let alone classifying these objects.

To improve this problem, this work mainly adopts a simple method that rotating the image with some degrees to make more objects could be recognized by YOLO. Also, using proposed bounding box mapping algorithm to map all predicted bounding boxes from rotated images back to original image. Then using Post-processing algorithm to eliminate redundant bounding boxes.

2.2. Post-processing Algorithm

At the last stage, YOLO adopts non-maximum suppression(NMS) [21] as post-processing algorithm to improve the network's performance.

NMS, as the name nom-maximum suppression implies, suppresses elements that are not maxima to search for local maxima. The general NMS algorithm is used to keep the bounding boxes with the highest scores and to eliminate redundant bounding boxes from predicted bounding boxes. For example, in object detection, the sliding window extracts numerous bounding boxes, and each bounding box will get a score after being classified by the classifier. But sliding the window can cause many bounding boxes exist and most of them intersect with other bounding boxes. At this time, you need to use NMS to select and keep the bounding boxes with highest scores (the probability of the object being the max) in those neighborhoods, and suppress those neighbor bounding boxes with low scores.

Recently, there is a new NMS algorithm, soft NMS [22], being proposed. Compared with standard NMS, the soft NMS changes the scores or confidences of bounding boxes, which the IoU between them and the bounding Box with highest score is greater than threshold, by multiplying a factor, instead of discarding these bounding boxes directly.

2.2.1. NMS

NMS is the most prevalent post-processing for object detection. When get bounding boxes from detection network, NMS firstly sort all boxes by detection confidence, and take the box with highest confidence as base box, then calculate the intersection of union (IoU) between other boxes and this base box. If iou between another box and base box is greater than threshold, usually set as 0.45, this box will be regarded as redundant box and will be eliminated. The pseudo code of NMS algorithm shown as Figure. 2-5.

Input : $\mathcal{B} = \{b_1, .., b_N\}, \mathcal{S} = \{s_1, .., s_N\}, N_t$
$\mathcal{B}$ is the list of initial detection boxes
$\mathcal{S}$ contains corresponding detection scores
N_t is the NMS threshold

begin
  $\mathcal{D} \leftarrow \{\}$
  while $\mathcal{B} \neq empty$ **do**
    $m \leftarrow$ argmax $\mathcal{S}$
    $\mathcal{M} \leftarrow b_m$
    $\mathcal{D} \leftarrow \mathcal{D} \bigcup \mathcal{M}; \mathcal{B} \leftarrow \mathcal{B} - \mathcal{M}$
    for b_i *in* $\mathcal{B}$ **do**
      if $iou(\mathcal{M}, b_i) \geq N_t$ **then**
        $\mathcal{B} \leftarrow \mathcal{B} - b_i; \mathcal{S} \leftarrow \mathcal{S} - s_i$
      end
    end
  end
  return $\mathcal{D}, \mathcal{S}$
end

NMS

Figure 2-5. The pseudo code of NMS algorithm

The result of NMS shown as Figure 2-6.

a) b)

Figure 2-6. Result of NMS, a) result before NMS b) result after NMS.

2.2.2. Soft NMS

Similar to NMS, when get bounding boxes from detection network, soft NMS firstly sort all boxes by detection confidence, and take the box with highest confidence as base box, then calculate the intersection of union (IoU) between other boxes and this base box. If iou between another box and base box is greater than threshold, this box' s score will be re-calculated as the formula

$$score_{new} = (1 - iou) * score$$

ranther than this box will be regarded as redundant box and will be eliminated. Figure 2-7 shows the pseudo code of soft NMS algorithm.

In normal case, the result of soft NMS is similar to, sometimes even same with that of NMS.

a) b)

Figure 2-7. Result of soft NMS, a) result before NMS b) result after soft NMS.

2.2.3. Limitation of NMS and soft NMS

Although NMS and soft NMS generally have a good performance on removing redundant bounding box, under the situation that objects in image are very close or overlapping, NMS and soft NMS usually eliminate the adjacent box which is correct bounding box because it has high IoU with the base box.

Besides, when two objects are too close to each other or are overlapping, the detection network sometime will detect two objects as one. In other word, the detection network will generate one big bounding box for these two objects, whereas we need two boxes in fact, to localize these two objects, but this big box is wrong and need to be removed by NMS or soft NMS. However, due to the existing of this big bounding box, the NMS and soft NMS always eliminate the right bounding boxes that covered by this big bounding box and keep this wrong big bounding box.

In the Figure 2-8, the left image shows us the result before NMS, form all bounding boxes in the left image, the NMS algorithm should keep the black bounding boxes and remove other redundant bounding boxes. However, from result of the right image, the right bounding boxes are removed and the wrong one is kept since it has higher score.

a) b)

Figure 2-8. Error result of NMS and soft NMS, a) result before NMS b) result after NMS

To ameliorate these problems, This Thesis proses a novel NMS algorithm called penalty NMS, which based on the different idea compared with NMS and soft NMS.

Chapter 3. Proposed Algorithm

3.1. Detection of rotated images

Normally, when objects in image are close or overlapping, it is difficult for YOLO network to detect and localize all objects accurately. However, even though some objects cannot be detected in given image, we can rotate image with arbitrary degree, then some objects can be detected whereas others cannot. Therefore, this work proposes a method that synthesize all detection results in original image and in rotated images, in other words, we keep the detection results in original image, and then map all detected bounding boxes back to original image.

For example, Figure 3-1 shows the detection results of YOLO v3 for a given image. For the given image, as two objects in the image are so close, even a little overlapping, that the YOLO network recognize them as one object. In other word, for the given image, we need two bounding boxes to localize the two objects, two bottles in the Figure 3-1, held by person in the image, whereas the YOLO network thinks of these two objects as one so that YOLO just generates one bounding box to localize these two objects. This is definitely wrong detection

results.

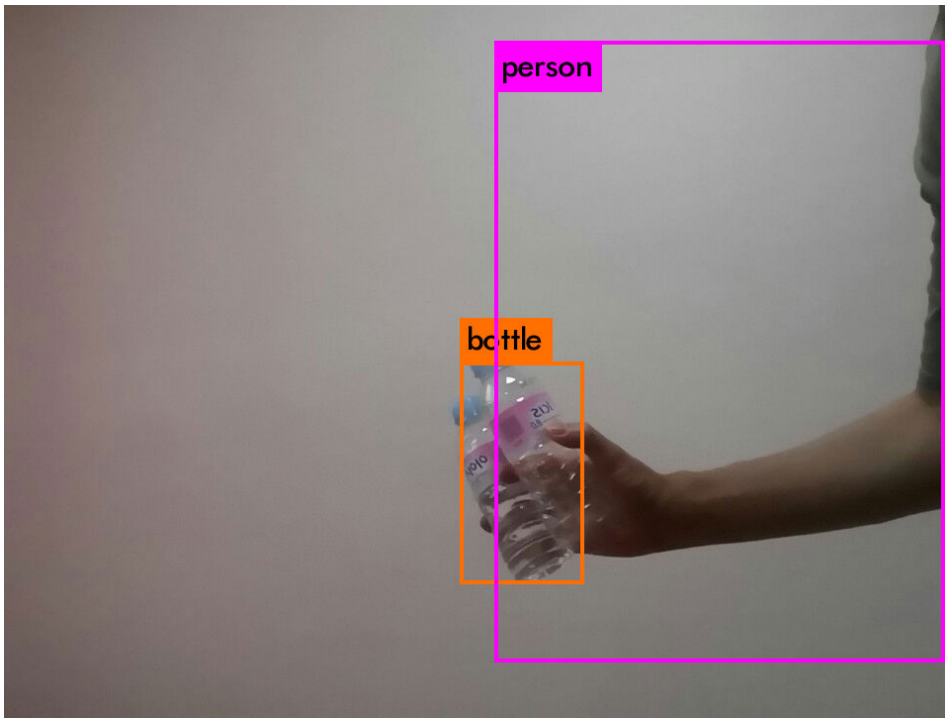
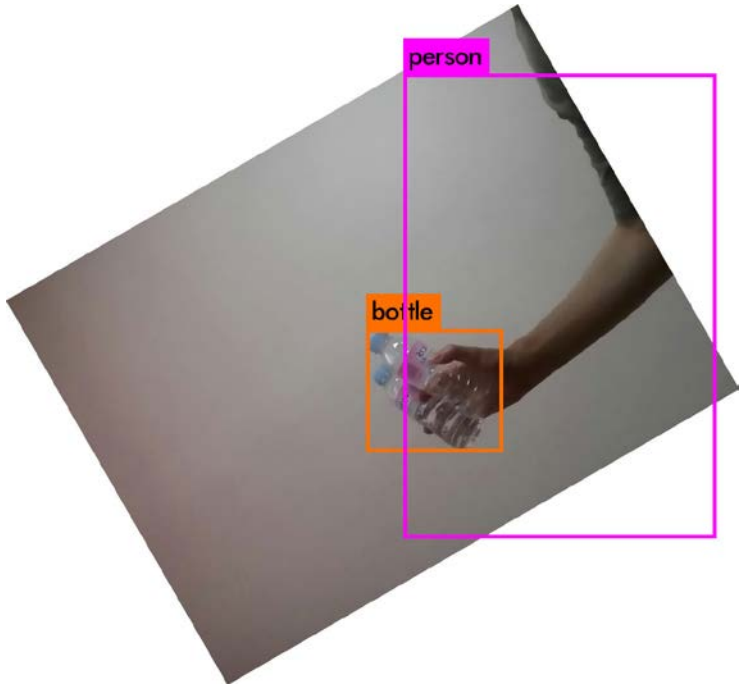
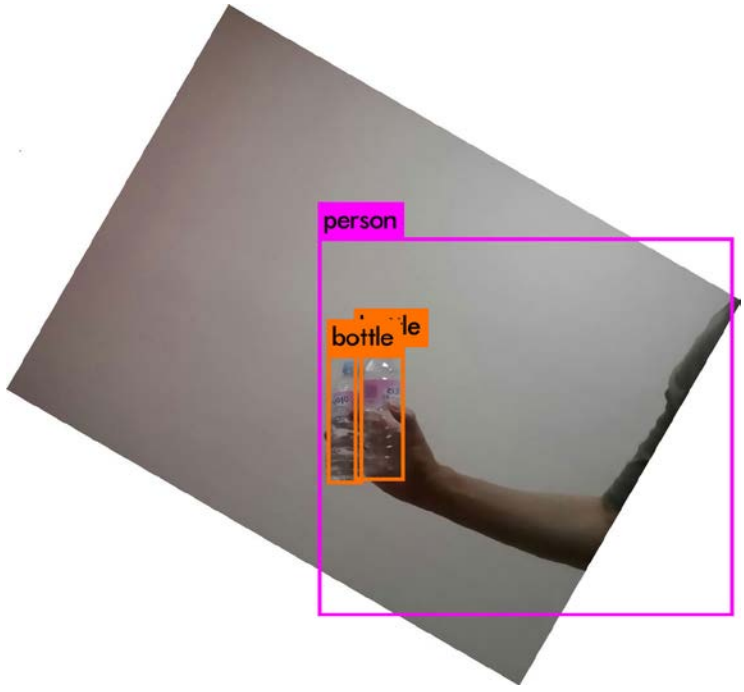


Figure 3-1. The detection results of YOLO v3

In the experiments, when the image is rotated with an arbitrary degree, one or two or all objects in the image can be detected and localized accurately. For example, when the image is rotated 30 degrees in counter-clockwise, the two objects are still recognized as one object. However, when the given image is rotated 30 degrees in clockwise, these two objects can be correctly and accurately detected and localized. Figure 3-2 shows the detection results of rotated images by YOLO v3.



a)



b)

Figure 3-2. The detection results of rotated images, a) counter-clockwise rotation in 30 degrees b) clockwise rotation in 30 degrees

Figure 3-3 shows one more example. There are two objects - cell phone and remote - in the image. However, for the given image, YOLO v3 just detect and localize the cell phone. When the given image is counter-clockwise rotated 30 degrees, both cell phone and remote can be detected accurately although remote cannot be detected in the clockwise rotated image.

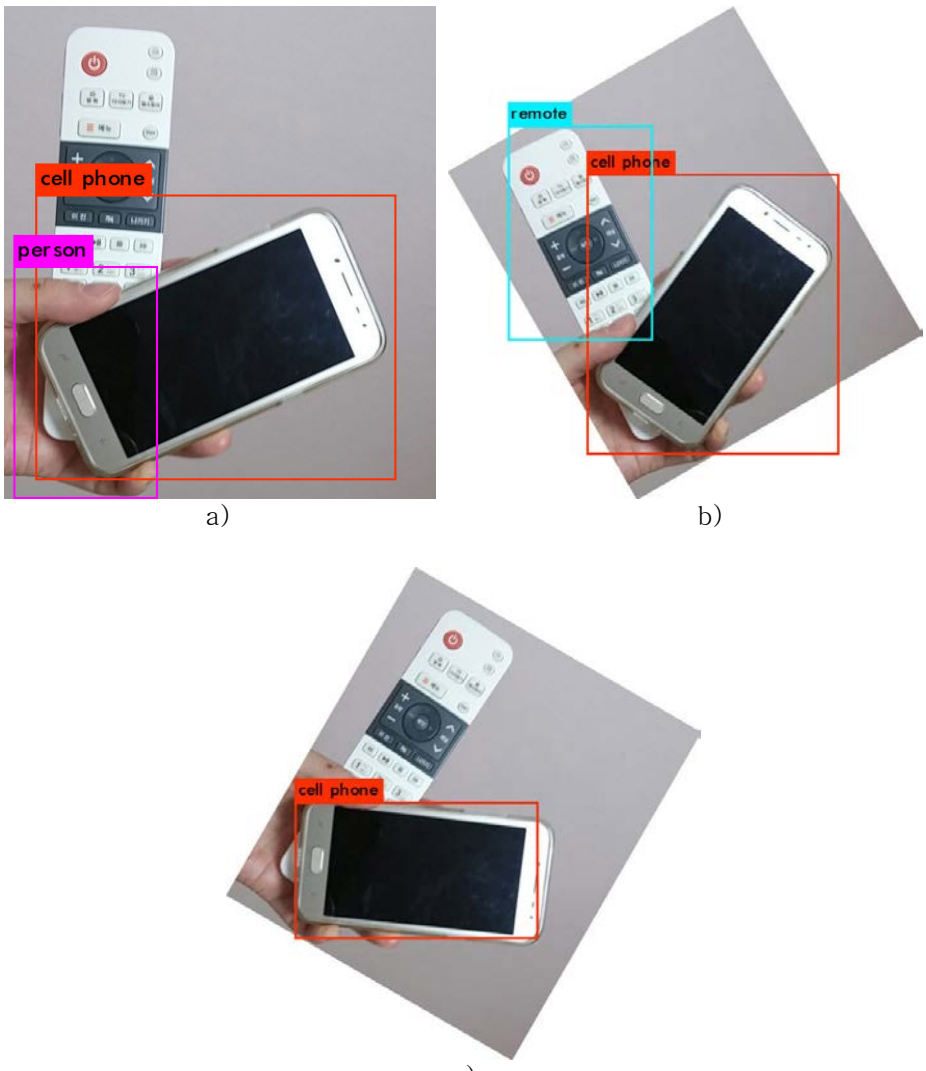


Figure 3-3. The detection results, a) given original image b) counter-clockwise rotation in 30 degrees c) clockwise rotation in 30 degrees

3.2. Bounding boxes mapping algorithm

Basically, people expect that when input the image to the YOLO network, we can get the correct and accurate bounding boxes used to localize all objects on the input image instead of on the rotated image. Hence, this work proposes a bounding boxes mapping algorithm that could map all bounding boxes back to original input image in a proper way.

Firstly, finding the center point of bounding box in the rotated image, using affine matrix to map the center point from rotated image back to original image. In other words, this step is mainly to find the corresponding point in original image for the center point of bounding box in rotated image. the affine matrix normally is:

$$\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

After finding the corresponding center point in original image, to make the bounding box surround the object more accurate, the next step is finding proper width and height for the object in original image since the width and height of bounding box in rotated image do not match that in original image.

no matter which direction the image is rotated, as long as the degree is fixed, the new width and height of rotated image are same. Therefore, if the image is rotated 30 degrees no matter counter-

clockwise or clockwise, we firstly rotate the bounding box with the same degree to get new width and height of bounding box in rotated image, which become larger.

$$\text{width}_{\text{new}} = height * |\sin\theta| + width * |\cos\theta|$$

$$\text{height}_{\text{new}} = width * |\sin\theta| + height * |\cos\theta|$$

Then multiplying the new width and height of rotated bounding box with a factor to resize the bounding box, or proportionally shorten the width and height of rotated bounding box to get the most proper width and height of bounding box for the object in original given image. in this algorithm, the ratio is compute as:

$$\text{ratio} = \frac{\sqrt{width^2 + height^2}}{\sqrt{width_{new}^2 + height_{new}^2}}$$

$$\text{width}_{\text{bb}} = width_{new} * ratio, \qquad height_{bb} = height_{new} * ratio$$

Figure 3-4 shows the bounding boxes mapping results.

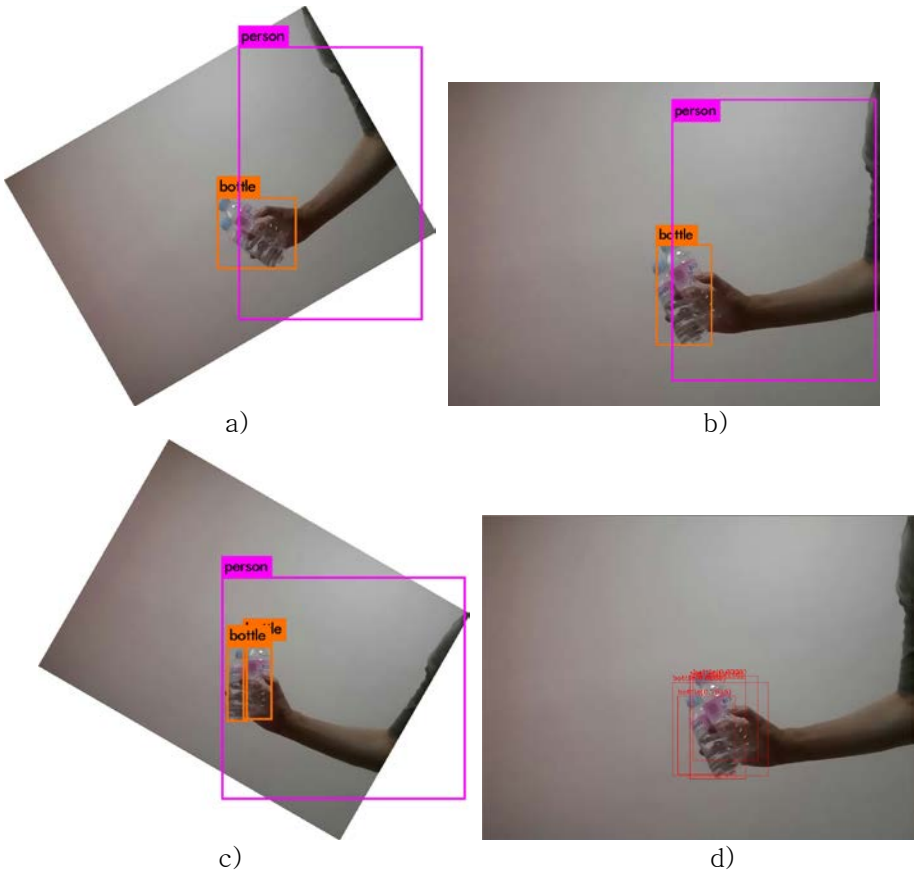


Figure 3-4. Detection results and bounding boxes mapping results, a) detection result of counter-clockwise rotated image b) original given image c) detection result of clockwise rotated image d) bounding boxes mapping results

3.2. Penalty NMS

Instead of keeping these bounding boxes with higher scores and discarding other bounding boxes with low scores. Penalty NMS first check if the box covers more than one object with same class, if that, re-calculate the score of this box, or if this bounding box covers one bounding box only, re-calculate the score of this covered bounding box, otherwise continue process on next box.

The penalty NMS includes three steps:

- Step 1: take the bounding box with highest score as base bounding box, check how many bounding boxes are covered by this base bounding box.
- Step 2: check how many covered bounding boxes are independent covered bounding box.
- Step 3: based on the numbers of independent covered bounding boxes, re-calculate score of covered bounding box or base bounding box.

Step 1: check covered bounding box

To check how many objects with same class, this work introduces intersection of self’ s area (IoSA) that used to check if one box is in another or not.

$$\text{iosa}(\text{box}_a, \text{box}_b) = \frac{\text{intersection area}(\text{box}_a, \text{box}_b)}{\text{box}_a\text{'s area}}$$

$$\text{iou}(\text{box}_a, \text{box}_b) = \frac{\text{intersection area}(\text{box}_a, \text{box}_b)}{\text{union area}(\text{box}_a, \text{box}_b)}$$

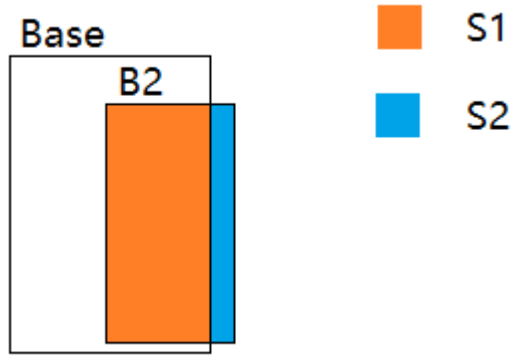


Figure 3-5. Example for IoSA,
S1 is intersection area and the area of B2 is S1 + S2

For example, in Figure 3-5, this work checks whether base bounding box covers bounding box B2 by calculating the intersection of self-area instead of check the vertexes and sides of base bounding box and B2. If the value of intersection of self-area between base bounding box and B2 is greater than threshold, then B2 is regarded as covered by base bounding box, marked as covered bounding box.

Step 2: check independent covered bounding box

This work defines independent covered bounding box that if a covered bounding box doesn’ t cover any other covered bounding box, on the other hand, if this covered bounding box covers some other covered bounding boxes, they are regarded as one independent covered bounding box.

If one box contains many objects belonging to same class, the next step is to check if these covered bounding boxes are independent covered bounding boxes, which means if they localize different objects.

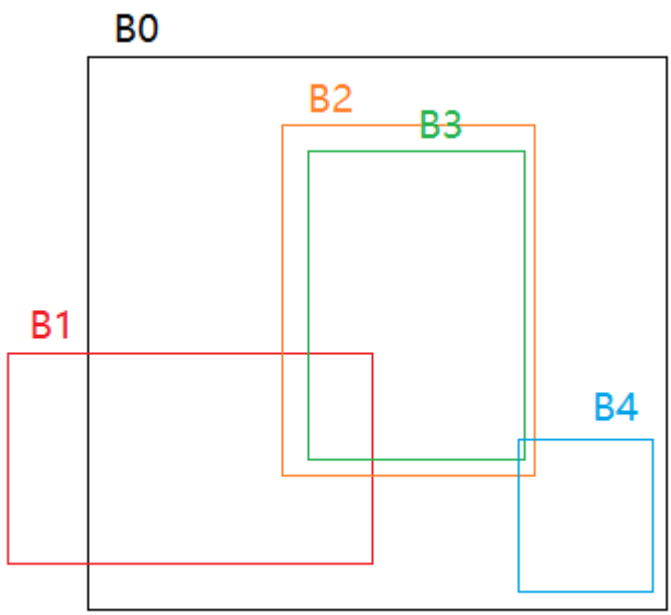


Figure 3-6. example for covered bounding box
Base bounding box: B0
Covered bounding box: B1, B2, B3, B4

For example, for the base bounding box B0, it covers four bounding boxes that are B1, B2, B3, B4 respectively. For these four covered bounding boxes, B1 and B4 are generated for localizing different objects whereas B2 and B3 localize the same object. In other word, B0 covers three objects, and for these three objects B1 take response of one, B4 covers one and B2 and B3 localize one. Hence B0 only covers three independent bounding boxes because B2 and B3 are regarded as one independent bounding box.

Step 3: re-calculate score

If the base bounding box covers one independent covered bounding box, this independent covered bounding box' s score will be re-calculated as the way soft NMS does.

$$score_{coveredBBox}^{new} = (1 - iou) * score_{coveredBbox}$$

If the new score of covered bounding box is less than threshold, this independent covered bounding box will be removed, otherwise continue to next bounding box

If the base bounding box covers more than one independent covered bounding box, the score of base bounding box will be recalculated as the formula

$$\text{score}_{\text{baseB}box}^{new} = score_{baseBbox} * \prod_{i=1}^{N} 1 - iou(baseBbox, ICBox_i)$$

If the new score of base bounding box is smaller than threshold, this base bounding box will be removed, otherwise continue to next bounding box.

The pseudo code of proposed penalty NMS shown as Figure 3-7.

Input: B = $\{b_1, \dots, b_2\}$, S = $\{s_1, \dots, s_N\}$
begin
    D = { }
    while B ≠ empty **do**
        $m = argmax\ S; M = B_m$
        for b_i in B **do**
            Check the numbers (N) of independent covered bounding boxes
            if N = 1 **do**
                $s_i = s_i * (1 - iou(M, b_i))$
                $D = D \cup M; B = B - M$
            if N > 1 **do**
                $s_m = s_m * \prod_{j=1}^{N} (1 - iou(M, b_i))$
        end
    end
    return D, S
end

Figure 3-7. Pseudo code of penalty NMS algorithm

Chapter 4. Experimental Results

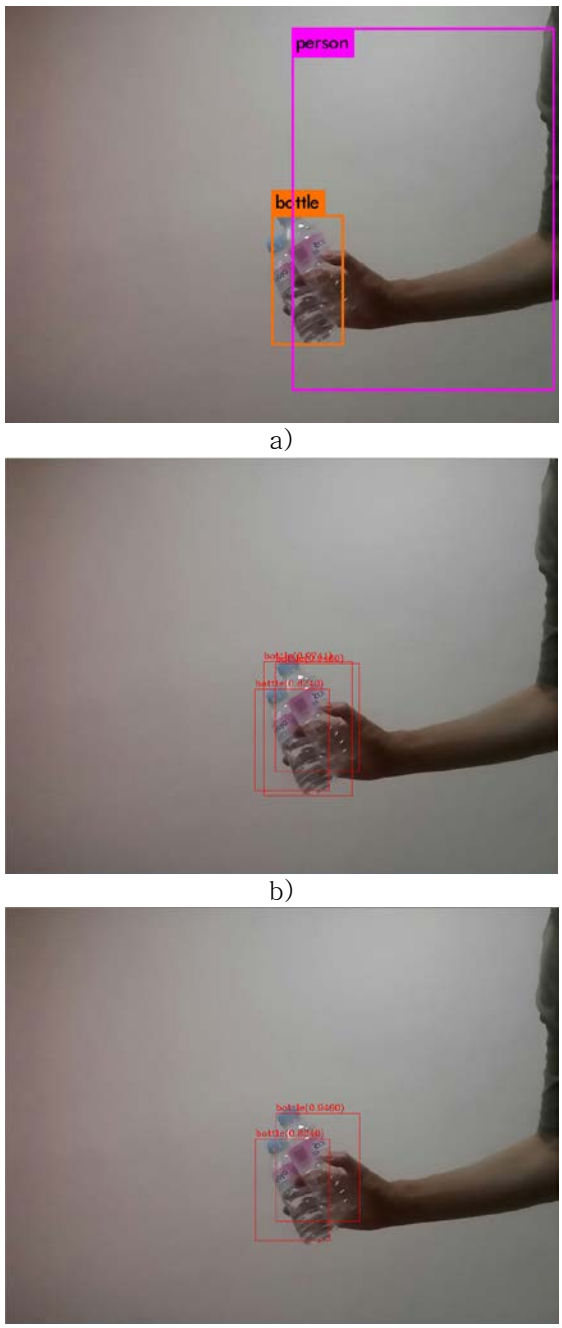
The proposed method adopted YOLOv3, which uses C programming language and CUDA-C GPU programming language, as main CNN prediction network. This fast and real-time network is suitable for daily life project.

To evaluate the performance of proposed algorithm, we took ten classes of objects from COCO dataset to record dataset for close and overlapping objects. These ten classes are 'phone', 'bottle', 'mouse', 'cup', 'remote', 'apple', 'orange', 'banana', 'toothbrush', 'spoon' respectively. All objects of these ten classes are small and many of them could be hold by hand simultaneously. Under this situation, the objects in hand are very close to each other and overlapped by hand and finger, and are sometime overlapped by other objects in hand.

4.1. Results Comparison

The table 4-1 shows the number accuracy, which means for the given image, how many predicted bounding boxes are correct and match to ground truth. In Figure 4-1, the number accuracy of a) is 0.5, that of b) is 0.67, and that of c) is 1. In other word, when computing number accuracy, we take the maximum numbers between the numbers of ground truth bounding boxes and the numbers of

predicted bounding boxes as total numbers of bounding boxes, rather than just directly take the numbers of ground truth as total numbers.



a)

b)

c)

Figure 4-1. Detection results of different approaches

	R-NMS	R-Soft NMS	R-Proposed
Dataset(1055 images)	31%	34.7%	52.6%
	NMS	Soft NMS	Proposed
Dataset(1055 images)	25.3%	27%	30.1%

Table 4-1. The number accuracy comparison of different approaches. R-NMS means using rotated images, bounding box mapping algorithm and NMS, likewise, R-Soft NMS means using rotated images, bounding box mapping algorithm and Soft NMS, and R-proposed mean.

The results of Table 4-1 obtained by using rotated images that rotate given image from -90 degrees to 90 degrees with step 15 degrees, which generates 13 rotated image in all. The Figure 4-2 shows how number accuracy changes with the rotation degree range and rotated images increase.

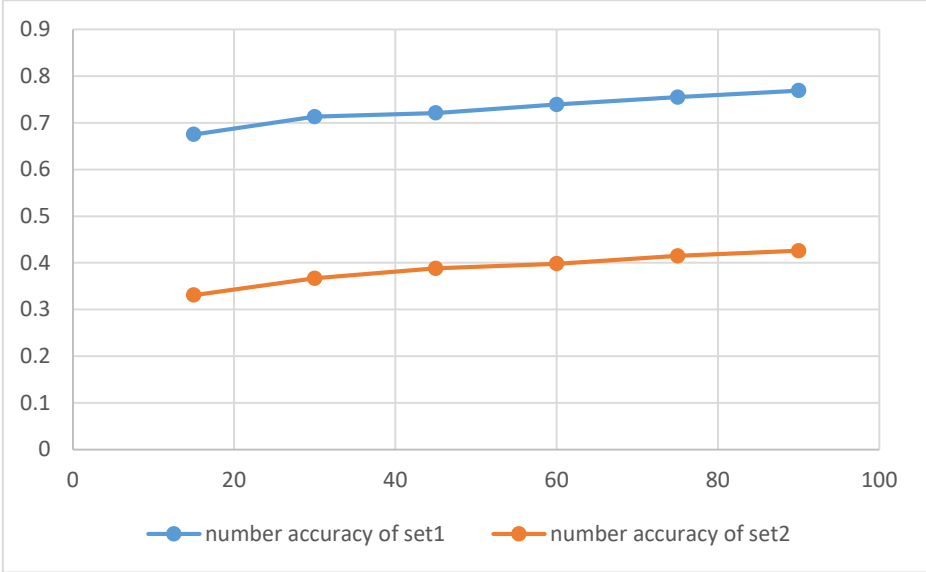


Figure 4-2. The relation between number accuracy and rotation degree range.

From the table 4-2, we can see that with the increase of rotation degree range, the number accuracy increase gradually. Since when numbers of rotated image grow, the CNN prediction network can generate more predicted bounding boxes. From these predicted bounding boxes, we could get the desired boxes for objects in given image and remove redundant boxes by proposed NMS.

Besides, table 4-3 shows the detection results on COCO dataset.

	AP	AP50	AP75	AR1	AR10	AR100
NMS	33.0	57.7	34.3	28.6	44.2	46.3
Soft NMS	33.5	56.5	36.1	28.6	48.0	51.7
Proposed	33.8	57.6	35.1	28.6	49.3	54.4

Table 4-2. detection results on COCO dataset (%).

4.2. Qualitative results examples

For the examples in Figure 4-3, a) is the result of YOLOv3, b) is the result of proposed method.

a)

b)

a)

b)

Figure 4-3. Qualitative example.

Chapter 5. Conclusion

This thesis proposes a pre- and post-processing method for improving the correctness rate on counting the numbers of objects in given image where the objects are close to each other and even overlapped by others or hand.

Normally, detection networks are only sensitive to the object that is perpendicular to horizon. When objects are rotated with some degrees, it is difficult for network to recognize the objects in given image. To solve this problem, this work adopts a pre-processing method that rotates input image with some degrees and then input the both rotated images and given images to detection network. Finally, this method increases the number accuracy by 0.185 on collected data.

On the other hand, because objects in image are close and overlapped, the detection network sometimes generates a big bounding box with high confidence covering two or more objects. Standard post-processing algorithm, Greedy NMS, will eliminate right boxes that contained in big box and remain the wrong big box. Proposed NMS commendably solves this problem.

Finally, this work collects around 1000 images contains close or overlapping objects as test dataset, the number accuracy of YOLOv3 on this dataset is 0.253, and that of proposed method is 0.526, increased by 0.273.

Bibliography

[1] C. Papageorgiou, M. Oren, and T. Poggio. A general framework for object detection. In International Conference on Computer Vision, 1998

[2] P. Viola, and M. Jones. Rapid object detection using a boosted cascade of simple features. In: CVPR (1). (2001) 511 – 518

[3] R. Lienhart, and J. Maydt. An extended set of Haar-like features for rapid object detection, in Proceedings of IEEE Conference on Image Processing (IEEE, 2002).

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In CVPR, 2005.

[5] Q. Zhu, S. Avidan, M. Yeh, and K. Cheng. Fast human detection using a cascade of histograms of oriented gradients, in CVPR, 2006.

[6] X. Wang, X. Han, and S. Yan. A HOG-LBP human detector with partial occlusion handling, in ICCV, 2009.

[7] Y. Pang, Y. Yuan, X. Li, et al. Efficient HOG human detection [J]. Signal Processing, 2011, 91: 773-781.

[8] P. Felzenszwalb, D. McAllester, D. Ramaman. A Discriminatively Trained, Multiscale, Deformable Part Model, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2008.

[9] R. Girshick, P. Felzenszwalb, D. McAllester. Object Detection with Grammar

Models, Neural Information Processing Systems (NIPS), 2011.

[10] P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan. Object Detection with Discriminatively Trained Part Based Models IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010.

[11] P. Felzenszwalb, R. Girshick, D. McAllester. Cascade Object Detection with Deformable Part Models IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010.

[12] P. Felzenszwalb, D. McAllester. Object Detection Grammars, University of Chicago, Computer Science TR-2010-02, February 2010.

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

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

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

[16] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, and S. E. Reed. SSD: Single Shot Multibox Detector. CoRR, abs/1512.02325, 2015.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation, in CVPR, 2014.

[18] R. Girshick. Fast R-CNN, in IEEE International Conference on Computer Vision (ICCV), 2015.

[19] S. Ren, K. He, R. Girshick, and J. Sun. Faster R-CNN: Towards Real-Time Object

Detection with Region Proposal Networks, in NIPS, 2015

[20] K. He, X. Zhang, S. Ren, and J. Sun. Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition, in European Conference on Computer Vision (ECCV), 2014.

[21] A. Neubeck, L. Van Gool. Efficient non-maximum suppression, in ICPR, 2006.

[22] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Improving Object Detection with One Line of Code. In Computer Vision (ICCV), 2017 IEEE International Conference on, pages 5562–5570. IEEE, 2017.

Abstract in Korean

Object detection은 Computer vision에서의 중요한 연구 분야로 영상에서 물체의 종류, 위치, 개수 등을 인식한다. 이러한 인식에는 CNN(Convolutional Neural Network)에 기반을 둔 방식이 널리 사용되고 있으며 전반적으로 만족할만한 성능을 보여준다. 하지만 전통적인 CNN 방법은 물체 영역을 rectangle 단위로 검출하며 rectangle 영역의 겹침 정도에 따라 독립적인 물체로의 판단 여부를 결정한다. 이러한 방법으로 인하여 물체의 겹침, 근접정도, 각도에 따라 물체의 개수를 정확하게 인식하는 데에는 한계가 존재한다.

이를 해결하기 위한 방법 중 하나는 training 과정에서 물체의 겹침이나 각도 등을 학습하게 한다. 하지만 이러한 방법은 dataset 구축과 학습에 상당한 시간을 필요로 하는 단점이 있다. 본 논문에서는 전통적인 CNN 인식 결과를 algorithm로 후처리 하는 방식을 제안한다. 물체의 각도에 따라 검출 영역이 다를 수 있다는 점에 착안하여 하나의 영상에 대하여 다양한 회전영상을 생성하여 물체의 영역을 검출하고 이 영역들의 확률과 상관관계 등을 종합적으로 판단하여 물체의 개수를 판단한다. 이 방법은 초기 성능 평가 결과 기존 방법 대비 향상된 성능을 보여준다.

주요어: object detection, non-maximum suppression(NMS), you look only once(YOLO).

학 번: 2017-20513