Trident SSD: A Trident Single-Shot Multibox Object Detector with Deconvolution

Xinlong Li, Xingwei Li*, Shaojie Guan, Jiating Jin and Yizhi Ge

College of Intelligence Science and Technology, National University of Defense Technology, Changsha 410073, China
Email: lxw_mmoml@nudt.edu.cn

Abstract. Scale variation is one of the key challenges in the object detection area, which limits the precision of detection methods like Single shot multibox detector (SSD). This paper proposes a detection method based on SSD, which focuses on handling scale variation and better detection performance of small objects. Our method, called TridentSSD, have an architecture with three branches, which are respectively responsible for detecting different scales of objects, solving the problem of scale variation while training. Then we augment small object branch with deconvolution module and feature fusion methods to improve precision, especially for small object detection. During training, we modify the original matching rules to generate training samples. Consequently, we can use objects of different scales to train the corresponding branches respectively. Finally, Experiments have done on both PASCAL VOC2007 and VOC2012 datasets. Results show that, with an input size of 300×300, our TridentSSD achieves better performance compared to the benchmark method SSD.

1. Introduction

In recent years, object detection methods based on deep learning have made great progress. These methods are mainly used in unmanned driving, multi-object tracking, scene understanding. Object detectors based on deep learning are roughly classified into two types: one-stage detectors and two-stage detectors. Two-stage detectors (e.g., Faster R-CNN [1]) first generate regions of interest (ROIs) by selective search or region proposal networks (RPN), and then classifies the ROIs and regresses bounding box using a convolutional neural network. In contrast, One-stage detectors, such as SSD [2] and YOLO [3], directly perform classification and regression without generating proposals. One-stage detectors require less computation and are faster at the cost of precision. Although deep learning object detectors have achieved wonderful performance, there are still many challenges to overcome for improving precision.

The occlusion problem is one of the main challenges that limit the precision of the object detectors. At present, there are not many effective methods to deal with the problem of occlusion. Wang et al. [4] proposed Repulsion Loss to suppress the occurrence of occlusion by making the boxes of prediction close to the ground truth (GT) and away from the other boxes. Shi et al. [5] partitioned the object and suppressing the confidence weight of the occluded part to improve the overall confidence of occlusion objects. These methods have achieved favourable results, but unfortunately, they are only suitable for pedestrian detection.

Low performance of small object detection is another challenge for increasing precision. The difficulty of detecting small objects lies in that the semantic information is weak on the lower-level feature maps, while on the high-level feature maps the scale of the feature corresponding to small objects...
is too small. So many methods improved performance of small target detection by multi-scale feature fusion, such as FPN [6], DSSD [7]. PyramidBox [8] uses context information to assist the detection of small objects, but currently only used on human faces.

Another challenge for improving precision is the large-scale variation of the training data. First the size variation of different object is large. Second the numbers of different size of objects are extremely imbalanced. Efforts have been done on this challenge. Kisantal et al. [9] alleviated the imbalance of training samples by enhancing small object data, thereby improving the detection precision of small object samples. The SNIP [10] algorithm improves the detection precision by building an image pyramid. Although this method significantly increases the amount of calculation, the improvement in precision is also very obvious. TridentNet [11] learns objects of different scales through three branches, and through the sharing of parameters between the branches, the method got a wonderful result.

In this paper, in order to improve the ability of one-stage object detector to deal with scale variation, inspired by TridentNet [11], we have designed a one-stage detector with a three-branch network structure, using three branches to detect different scales of objects. During training, we use the objects of different scales to train the corresponding branches to reduce the impact of large difference of the object size in the image. And it also eases the impact of the imbalance between numbers of training samples from different scales. In order to optimize the one-stage object detector for detecting small objects, we add a deconvolution module to the small object branch. Through deconvolution, high-level features are upsampled and feature fusion is performed, so that higher-level semantic information can be used for small object detection to improve performance. In order to make the corresponding branch trained with the object of the corresponding scale, a new sample generation method is designed, which is used in conjunction with our training. Finally, through experiments, we select the most suitable sample scale range and the feature fusion method, and then use the optimal size range and feature fusion method to test our model.

The key contributions of our work are as follows:

- We propose a one-stage object detector with three branches that are respectively responsible for detecting objects in three different scale ranges. Multi-scale feature maps are designed in each branch to avoid large scale gap and imbalance of object number from different scales.
- We introduce deconvolution module in the small object branch and utilize multi-scale feature fusion to improve the detection precision of small objects.
- During training, we propose a new method of sample generation, which is used to match our branch structure network and select a suitable scale range through experiments.
- Experiments include timing and precision analysis on our models on the PASCAL VOC2007 [12] and VOC2012 data sets comparing with the benchmark method.

2. Related works

2.1. Deep Object Detection

Object detectors have achieved outstanding results in both speed and precision by introducing convolutional neural networks. A host of efforts have been done to improve the inference speed of detectors. The methods used to improve the speed include sharing feature maps (e.g., Faster R-CNN [1] and RFCN [13]), replacing traditional methods with deep learning methods (e.g., RPN [1]), and using a more lightweight backbone network. On the other hand, also many methods have been proposed to improve the detection precision. FPN [6] proposed feature pyramid network (FPN) for multi-scale feature fusion. FSSD [14] applied feature fusion method to one-stage detectors and improved small object detection performance. Bodla et al. [15] changes the original non-maximum suppression (NMS) method to improve the detector's ability to process occlusions. Lin et al. [16] proposed a more comprehensive loss calculation method to handle the problem of imbalance between positive and negative training samples by reducing the loss weight of negative samples.
2.2. Methods Handling Scales Variation
As a challenging task of object detection task, the large scale difference between the objects affects the performance of the object detector. The image pyramid of size is the most commonly used method to improve the detection precision, especially for objects from small and large scales. Based on the theory of image pyramids, SNIP [10] proposes a scale normalization model, and then multi-scale normalizes the scale of the object to a certain range for training. TridentNet [11] constructs convolutional branches with different receptive fields by using convolutional layers with different dilation and training with objects from different scales. SSD [2] adapt to detect objects of different scale by using multi-scale feature maps. FPN proposed the feature pyramid networks and used a top-down channel for feature fusion and enrich semantic information for low-level feature map. PanNet [17] adds a bottom-up channel on the basis of FPN to enhance the feature information of all layers to improve the detection performance.

2.3. Deconvolution
Deconvolution, also known as transposed convolution, is one of the methods for upsampling. Due to its learnability, deconvolution usually performs better than other upsampling methods such as linear interpolation. It has been widely used in semantic segmentation to form a decoder, such as FCN [18], which used deconvolution to upsample the heat map to the input size for pixel-level classification. In object detection field, deconvolution was used to expand the size of high-level feature maps for feature fusion. DSSD [7] upsampled the high-level feature maps through deconvolution and fused feature maps to enhance low-level features, helping improve the detection accuracy.

3. Method
3.1. Trident Network Architecture
The trident network architecture is shown in figure 1. The performance of the SSD is limited by the detection of small objects. The inevitable problem of using a single branch network structure for object detection is that the number of objects from different scales in the training set is uneven, and the proportion of small objects is always small. This will lead to a decrease in the detection performance for small objects, when the features are not sufficiently trained. In order to handle the problem of scales variation, we adopt a network architecture with three branches. Each branch is only responsible for the detection of objects within a certain scale range, and this greatly reduces the impact of scales variation.

Multi-scale feature maps: The network structure of each branch is similar to the structure of SSD [2]. Process of each branch can be divided into two steps: feature extraction; classification and bounding box regression. In each branch of our network structure, we have designed multi-scale feature maps for detecting objects of different sizes in the range of each branch, and default boxes are set on these feature maps. In theory, the more the scale of the feature maps, the more adequate the detection. However, too many feature maps in each branch will lead to excessive overlap of default boxes, increasing the amount of calculation and also interfering with the detection. Finally, we decide that the size of the feature maps used for classification and regression is $\{38, 19, 10\}$ for the small object branch, and $\{10, 5, 3\}$ for the medium object branch, and $\{3, 1\}$ for large object branch.

3.2. Deconvolution and Feature Fusion Module
The reason why SSD has poor detection performance for small objects is that the lower-level feature maps contain more texture and colour features, but less semantic information. In order to utilize semantic information of high-level features for small object detection, we design learnable deconvolutional modules for feature fusion. In order to improve feature fusion while avoiding excessive calculation, we only introduced deconvolution in the small object branch. It is not used in the large object branch and the medium object branch, because their feature maps already have enough semantic information. Its specific structure is shown in figure 2.
Figure 1. Networks of TridentSSD on VGG16. The module in dotted frame is VGG16. The numbers on the side of the feature maps indicate their number of channels. Feature maps produced by deconvolution are painted dark.

Figure 2. Layers of feature fusion modules: (a) the fusion module for element-wise sum and product. (b) the fusion module for concatenate. These three modules have the same input and output size.

The feature map obtained by deconvolution is merged with the original convolution feature map. We propose three models of feature fusion, as shown in figure 2, including element-wise sum, element-wise product and concatenate. Finally, we tested these three methods separately. For detailed test results, see section 4.2.

3.3. Sample Generation

Our sample generation method for training is the modification based on the original method used by SSD. The original method matches all default boxes with all ground truth (GT) boxes. The original
sample generation method matches each GT box with the default boxes which are best overlapped with the GT box or have Jaccard overlap larger than a threshold (e.g. 0.5). The difference is that we designed a network with three branches. If we use all GT boxes to match the default box of all the branches, it will cause one problem: the result may produce wrong matching, the scale of matched GT boxes may not in the scale range of the certain branch. This against the original intention for avoiding scale variation.

In order to solve the problem, we modify the match method so default boxes of three branches are matched with GT boxes respectively in three times. And for each branch, only the GT box in the scale range is considered. This matches the positive sample. In order to determine the most suitable training sample scale range for each branch, we selected three sets of scale ranges and tested them separately, more details in section 4.2. After samples are all labelled, to prevent the large quantitative variation between positive and negative samples used for loss calculation, hard negative mining (HNM) method is applied to keep the ratio of positive and negative samples at 1:3. The selection of negative samples is based on their confident loss. This method greatly influences the precision of one-stage detectors but not two-stage detectors which filtering out most of the negative samples due to the RPN network. The results obtained are used to calculate the loss and back propagation.

4. Experiments and Results

4.1. Settings and Environments

The picture shows the complete structure of our TridentSSD. The basenet we use is VGG16 [19], which is pre-trained on the ILSVRC CLS-LOC dataset [20]. Except for VGG layers, all network parameters are initialized using the Xavier method. More details are the same as the SSD algorithm. The feature maps we select are deconv1, deconv2, deconv3 as small branches, conv10_2, conv11_2, conv12_2 as medium branches, and conv15_2, conv16_2 as large branches. We set a series of default boxes on these feature maps, the sizes of these boxes are the same as SSD. The aspect ratio of default boxes satisfies that $i \in \{1, 2, 3, \frac{1}{2}, \frac{1}{3}\}$ when $i = 1$ there are two different sizes $S_1$ and $S_2$, when $i$ is other values, their area is $S_i$. For deconv3, conv12_2, conv15_2, and conv16_2, we only set 4 boxes with the ratio in $\{1, 2, \frac{1}{2}\}$, and for the other feature maps, we set 6 boxes. So 9368 boxes are set in total.

We fine-tune our model using SGD, 0.9 momentum, 0.0005 weight decay, and batch size 16 (largest size supported by our GPU). The initial learning rate and learning rate decay policy vary slightly in each experiment. All of our experiments are carried out on NVIDIA GeForce 1660Ti 6G and with a single thread on Intel i7 9750H@2.6GHz.

4.2. Ablation Study

Our proposed TridentSSD requires a suitable object scale range. As showing in table 1, we propose three scale ranges for comparing. The first one is called full range (FR). For each branch, we use all objects for matching to generate positive samples regardless of their size. The second one is used by TridentNet [11], and we call it range of TridentNet (RT). The last one is set according to the size range of the default box on the feature maps of each branch, which we call the range of boxes (RB).

In order to test the effects of these three ranges, we train our network on PASCAL VOC2007 trainval. In this experiment the feature fusion method we adopt is element-wise sum. We use the learning rate $7 \times 10^{-4}$ for the beginning 40k iterations, then $7 \times 10^{-5}$ and 10k $7 \times 10^{-6}$ for 10k iterations respectively. Finally, we test our model on PASCAL VOC2007 test. The precision results are shown in table 1. Result shows that it is most effective scale range is RB, which get a 71.4% mean average precision (mAP). RB is 1.6% better than RT and 2.9% better than FR. All of our methods get better performance compared to the well-known SSD. That means our trident architecture and deconvolution module does work. Among our methods, FR gets the worst result, it means our method benefits from scale-aware training. So, we select the object scale RB in subsequent experiments.
In order to pick out the best feature fusion method for the small object branch among the three methods proposed in section 3.2, we evaluate these methods on PASCAL VOC2007. The training and test data and training process are the same as those used in the scale range test. Our three feature fusion methods are element-wise sum, element-wise prod and concatenate, and their results are shown in the following table 2. First, all of our three methods get better results compared to SSD. Second, we can conclude that the performance gap among the three methods is small, and ‘cat’ method get the best mAP of 72.0%, and it is 0.3% and 0.6% better than ‘prod’ and ‘sum’ respectively. We believe that such a small performance gap is not enough for us to select the best one among them, so in the following experiments we evaluate all these three methods respectively.

**Table 1. Ablation study: Effects of different scale ranges compared to SSD on VOC2007 test. Data: 07: VOC2007 trainval for training.**

| Method       | Data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow |
|--------------|------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| SSD [2]      | 07   | 68  | 73.4 | 77.5 | 64.1 | 59.0 | 38.9   | 75.2| 80.8| 78.5| 46.0  | 67.8|
| Ours + FR + sum | 07  | 68.5| 73.5 | **80.1** | 65.7 | 56.0 | 32.6   | 79.8| 78.6| 83.5| 47.5  | 72.3|
| Ours + RT + sum | 07 | 69.8| **77.5** | **80.1** | **69.0** | 63.4 | **41.33** | 79.9| **81.9**| 80.4| **52.3** | 67.9|
| Ours + RB + sum | 07 | **71.4** | 75.5 | **80.1** | 68.7 | **65.2** | 40.0 | **80.1** | 81.6| **83.6** | 51.4| **74.3** |

**Table 2. Ablation Study: Effects of different feature fusion modules compared to SSD on VOC2007 test. Data: 07: VOC2007 trainval for training.**

| Method       | Data | mAP | areo | bike | bird | boat | bottle | bus | car | cat | chair | cow |
|--------------|------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| SSD [2]      | 07   | 68  | 73.4 | 77.5 | 64.1 | 59.0 | 38.9   | 75.2| 80.8| 78.5| 46.0  | 67.8|
| Ours + RB + sum | 07  | 71.4| 75.5 | 80.1 | 68.7 | **65.2** | 40.0   | 80.1| 81.6| 83.6| 51.4  | 74.3|
| Ours + RB + prod | 07 | 71.7| **77.6** | 82.4 | 70.1 | 63.7 | **41.9** | 79.5| 81.4| **84.5** | 52.1 | 76.3|
| Ours + RB + cat | 07 | **72.0** | 75.1 | **83.2** | **70.6** | 63.8 | 41.3 | **81.8** | **82.3** | 83.8 | **53.0** | **78.2** |

4.3. PASCAL VOC2007

The network structure parameters we used are the same as those used in 4.2, and we chose the third size range (RB) for each branch, which we chose. The training dataset we used in this experiment is a combination of VOC2007 trainval and VOC2012 trainval (16551 images), and the test set is VOC2007 test (4952 images). While training, we train our model for the first 80k iterations with the $10^{-3}$ learning rate, and then continue training for 20k iterations with $10^{-4}$ and $10^{-5}$. Table 3 shows the results of our TridentSSD with three different fusion methods comparing with some famous onestage detectors. It can
be seen from table 3 that ‘our + sum’ performances best among our three methods. When the input size is 300×300, the best mAP of our method is 4.1% higher than SSD, reaching 78.4%. We are also more accurate than RON384, Faster R-CNN and YOLOv2 even though they used larger input size.

4.4. PASCAL VOC2012
In the PASCAL VOC2012 task, the network settings are the same as those used in section 4.2. The difference is that the training set is the combination of VOC2007 trainval and VOC2007 test and VOC2012 trainval (21503 images in total) and the test set is VOC2012 test (10991 images). Compared with the VOC2007 task, the test set is increased to twice of the former, so this task is more challenging. During the training, we first used a learning rate of $10^{-3}$ to train 100k iterations, then used a learning rate of $10^{-4}$ to train 40k iterations and then used a learning rate of $10^{-5}$ to train 25k iterations. In table 4, we compare our method with the famous one-stage detectors. A similar result is got as VOC2007 task. Only small performance gaps exist between three fusion methods, and ‘Ours + sum’ performance best. Our methods are far more accurate than SSD, YOLO and Faster R-CNN reaching a mAP of 76.1%, and also outperform RON with smaller input size.

4.5. Inference Time
Before testing we can expect that our method is slower than SSD in many ways. First, our network contains more layers, which increase the amount of calculation. Second, we use more default boxes (9368 vs. 8732), which will increase the time for prediction and NMS, especially the latter. When testing, we use the batch size 1. Our timing result is shown in table 5 compared to some fast detectors. As expected, our TridentSSD is slower than SSD but much Faster R-CNN and DSSD. Considering our testing processes are carried out on our GPU, we believe better results can be reached on Titan X.

**Table 3.** PASCAL VOC2007 Test Detection Results. The number after the method indicates the size of the input. 07+12: VOC2007 trainval + VOC2012 trainval. Detail category precision is ungiven in YOLOv2 while their improved vision YOLOv3 has no result on this dataset. sum, prod, cat: respectively corresponding to three fusion methods mentioned above.

| Method       | Data | basenet | mAP | areo | bike | bird | Boat | bottle | bus | car | cat | chair | cow |
|--------------|------|---------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| Faster~600 [1] | 07+12 VGG16 | 73.2 | 76.5 | 79.0 | 70.9 | 65.5 | 52.1 | 83.1 | 84.7 | 86.4 | 52.0 | 81.9 |
| RON384++ [21] | 07+12 VGG16 | 77.6 | 86.0 | 82.5 | 76.9 | 69.1 | 59.2 | 86.2 | 85.5 | 87.2 | 59.9 | 81.4 |
| YOLOv2 352 [3] | 07+12 DarkNet | 73.7 | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| SSD300 [2] | 07+12 VGG16 | 74.3 | 75.5 | 80.2 | 72.3 | 66.3 | 47.6 | 83.0 | 84.2 | 86.1 | 54.7 | 78.3 |
| Ours + sum | 07+12 VGG16 | **78.4** | 81.0 | 84.4 | 78.0 | 72.5 | 50.9 | **86.4** | **86.2** | **88.7** | 62.5 | 83.9 |
| Ours + prod | 07+12 VGG16 | 78.0 | 78.2 | **84.7** | **79.2** | **72.9** | 52.1 | 85.5 | 85.3 | 88.4 | 62.2 | **84.6** |
| Ours+ cat | 07+12 VGG16 | 78.2 | **81.5** | 84.3 | 76.3 | 71.4 | 51.4 | 85.1 | 85.5 | 87.8 | **63.7** | 82.5 |

| Method       | Data | basenet | mAP | areo | bike | bird | Boat | bottle | bus | car | cat | chair | cow |
|--------------|------|---------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| Faster~600 [1] | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| RON384++ [21] | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| YOLOv2 352 [3] | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| SSD300 [2] | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Ours + sum | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Ours + prod | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |
| Ours+ cat | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- | -- |

Table 3: PASCAL VOC2007 Test Detection Results.
Table 4. PASCAL VOC2012 test detection results. The number after the method indicates the size of the input. 07++12: VOC2007 trainval+VOC2007 test + VOC2012 trainval. sum, prod, cat: respectively corresponding to three fusion methods.

| Method      | Data basenet | mAP | area | bike | bird | boat | bottle | bus | car | cat | chair | cow |
|-------------|--------------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| Faster ~600 [2] | 07++12 VGG16 | 68.4 | 82.3 | 78.4 | 70.8 | 52.3 | 38.7   | 77.8| 71.6| 89.3| 44.2  | 73.0 |
| RON384++ [21] | 07++12 VGG16 | 75.4 | 86.5 | 82.9 | 76.6 | 60.9 | 55.8   | 81.7| 80.2| 91.1| 57.3  | 81.1 |
| YOLOv2 352 | 07++12 DarkNet | 73.4 | 86.3 | 82.0 | 74.8 | 59.2 | 79.8   | 76.5| 90.6| 52.1| 78.2  |      |
| SSD 300 [2] | 07++12 VGG16 | 72.4 | 85.6 | 80.1 | 70.5 | 57.6 | 46.2   | 79.4| 76.1| 89.2| 53.0  | 77.0 |
| Ours + sum | 07++12 VGG16 | **76.1** | **89.4** | **85.0** | **75.5** | **62.1** | 50.6 | 82.8  | 77.2 | 91.8 | 57.8 | **80.8** |
| Ours + prod | 07++12 VGG16 | 75.9 | 87.7 | 84.0 | 75.3 | **62.1** | 51.4 | 82.5  | 77.2 | 92.0 | 57.0 | 80.5  |
| Ours + cat | 07++12 VGG16 | 75.9 | 88.4 | 83.4 | 75.2 | 60.9 | 51.3   | **82.8** | **77.6** | **92.2** | **57.9** | 80.6  |

Method Data basenet mAP area bike bird boat bottle bus car cat chair cow

| Method      | Data basenet | mAP | area | bike | bird | boat | bottle | bus | car | cat | chair | cow |
|-------------|--------------|-----|------|------|------|------|--------|-----|-----|-----|-------|-----|
| Faster ~600 [2] | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| RON384++ [21] | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| YOLOv2 352 | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| SSD 300 [2] | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| Ours + sum | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| Ours + prod | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |
| Ours + cat | --              | --  | --   | --   | --   | --   | --     | --  | --  | --  | --    | --  |

Table 5. Comparison of timing results on VOC2007 test task.

| Method      | Basenet | mAP | FPS | Boxes | GPU | Input resolution |
|-------------|---------|-----|-----|-------|-----|------------------|
| Faster R-CNN [1] | VGG16   | 73.2 | 7   | ~6000 | Titan X | ~600×1000 |
| SSD [2]     | VGG16   | 74.3 | 46  | 8732  | Titan X | 300×300  |
| DSSD [7]    | ResNet-101 | 78.6 | 9.5 | 17080 | Titan X | 321×321  |
| Ours        | VGG16   | 78.4 | 26.2| 9368  | 1660Ti  | 300×300  |

4.6. Visualization
In figure 3, we show some result images of our model compared to SSD. The samples are from VOC2007 test dataset. The model used is the model trained on PASCAL VOC2007 and VOC2012 trainval in section 4.3. In figure 3a, we show examples with large scale variation. From the results we can conclude that SSD always miss small objects when scales vary largely between objects, and our method performs much better. In figure 3b are the examples containing small object. Obviously, SSD tends to miss or misdetect some objects when objects are small, and our TridentSSD gets better results.
Figure 3. Detection examples on PASCAL VOC2007 test. (a): Examples that contain objects of large scales variation. (b): Examples of small objects detecting. For each pair, the left example is from SSD, and the right one is from TridentSSD. We only show detections with scores higher than 0.6.

5. Conclusion
We propose a one-stage detector with a three-branch network structure. This structure can help eliminate the problems caused by the imbalance of the data set by training each branch with the corresponding scales of images. We introduce a deconvolutional feature fusion module to the small object branch and fuse high-level semantic information to improve the detection performance of small objects. Then we modified the sample generation method to make different scales of samples matched with our branches for scale-aware training. Finally, experiments are done on PASCAL VOC. The quantitative and qualitative results prove that our TridentSSD outperforms the benchmark method SSD.
Acknowledgement
We’d like to thank professors in our research group for helpful discussions. We also thank the researchers in object detection field for all the good result we have referred.

References
[1] Ren S, He K, Girshick R and Sun J 2015 Faster R-CNN: Towards real-time object detection with region proposal networks IEEE Transactions on Pattern Analysis and Machine Intelligence 39 (6) 1137-1149.
[2] Liu W, Anguelov D, Erhan D, Szegedy C, Reed S, Fu C-Y and Berg A C 2016 Ssd: Single shot multibox detector European Conference on Computer Vision pp 21-37.
[3] Redmon J and Farhadi A 2017 YOLO9000: Better, faster, stronger Proc. of the IEEE Conference on Computer Vision and Pattern Recognition pp 7263-7271.
[4] Wang X, Xiao T, Jiang Y, Shao S, Sun J and Shen C 2018 Repulsion loss: Detecting pedestrians in a crowd IEEE Conference on Computer Vision and Pattern Recognition pp 7774-7783.
[5] Zhang S, Wen L, Bian X, Lei Z and Li S Z 2018 Occlusion-aware R-CNN: detecting pedestrian in a crowd European Conference on Computer Vision pp 637-653.
[6] Lin T-Y, Dollár P, Girshick R, He K, Hariharan B and Belongie S 2017 Feature pyramid networks for object detection IEEE Conference on Computer Vision and Pattern Recognition pp 2117-2125.
[7] Fu C-Y, Liu W, Ranga A, Tyagi A and Berg A C 2017 DSSD: Deconvolutional single shot detector.
[8] Tang X, Du D K, He Z and Liu J 2018 Pyramidbox: A context-assisted single shot face detector European Conference on Computer Vision pp 797-813.
[9] Singh B and Davis L S 2018 An analysis of scale invariance in object detection snip IEEE conference on Computer Vision and Pattern Recognition pp 3578-3587.
[10] Li Y, Chen Y, Wang N and Zhang Z 2019 Scale-aware trident networks for object detection.
[11] Everingham M, Gool L V, Williams C K I, Winn J and Zisserman A 2010 The Pascal visual object classes (VOC) challenge International Journal of Computer Vision 88 (2) 303-338.
[12] Dai J, Li Y, He K and Sun J 2016 R-FCN: Object detection via region-based fully convolutional networks Neural Information Processing Systems 379-387.
[13] Cao G, Xie X, Yang W, Liao Q, Shi G and Wu J 2018 Feature-fused SSD: fast detection for small objects Ninth International Conference on Graphics and Image Processing 2017 106151E.
[14] Bodla N, Singh B, Chellappa R and Davis L S 2017 Soft-NMS-improving object detection with one line of code IEEE International Conference on Computer Vision pp 5561-5569.
[15] Lin T-Y, Goyal P, Girshick R, He K and Dollár P 2017 Focal loss for dense object detection IEEE International Conference on Computer Vision pp 2980-2988.
[16] Yang J, Fu X, Hu Y, Yue H and Paisley J 2017 PanNet: A Deep Network Architecture for Pan- Sharpening IEEE International Conference on Computer Vision.
[17] Shelhamer E, Long J and Darrell T 2017 Fully convolutional networks for semantic Segmentation IEEE Computer Society.
[18] Simonyan K and Zisserman A 2014 Very deep convolutional networks for large-scale image recognition Computer Science.
[19] Russakovsky O, et al. 2015 ImageNet Large Scale Visual Recognition Challenge International Journal of Computer Vision 115 (3) 211-252.
[20] Kong T, Sun F, Yao A, Liu H, Lu M and Chen Y 2017 RON: Reverse connection with objectness prior networks for object detection.