# Small Object Detection Survey Paper

**Junheng Wang, Zhengjiang He, and Haomin Li**
{j765wang, z75he, h439li}@edu.uwaterloo.ca
University of Waterloo
Waterloo, ON, Canada

## Abstract

Object detection is a fundamental and important problem in computer vision. Some promising results have already been achieved in regular sized detection datasets, but the performance on small objects is still far from satisfactory. The poor performance on small objects sometimes can lead to really severe problems in some real world projects, such as self-driving, surveillance applications. In this report, we reviewed most of related research papers on small object detection and summarized current progress and potential improvements in the future. We found that most of the effective improvements are made via one of the three approaches: (i) Augment the dataset to have larger sample of small objects. (ii) Improve the training scheme and loss function to produce better model performance. (iii) Modify the existing model architecture to overcome the model bottlenecks.

## Introduction

Object detection has been one of the most fundamental problems that many computer vision researchers are trying to solve. There are quite a lot of deep learning approaches such as Fast RCNN, YOLO that already produce state-of-the-art detections. However, detecting small objects which occupy less than 1% of the image space is still an outstanding problem to solve.

There are three main reasons why detecting small objects would be considered as a difficult problem to solve in computer vision studies. First of all, most of the images have limited input resolution. If the images themselves are in a low resolution, then small objects may only be represented by a few pixels. With so limited information from the images, even humans would not perform well enough. In addition, most of the computer vision architectures scale the input size to a fixed size which is usually very small. This may reduce the image size further. Moreover, Many computer vision models use pooling layers which may remove features from the images. This would be an essential problem if some features of the small objects are removed after pooling.

While detection on small objects may not always be necessary, it could be typically crucial to some real world problems. For example, the vehicle perception models for self-driving requires extremely accurate detection behaviors on small objects, such as traffic lights and so on. Any failures of traffic lights detection can cause severe problems. In addition, there is a huge market potential in surveillance applications, such as facial recognition by security camera, parking lots monitors and so on. Detection of small objects, including human faces and parking spots is currently a bottleneck of such surveillance applications.

In this report, we focus on the first research question, summarize the existing papers that tried to tackle this problem and suggest some potential improvements in the future. As object detection becomes more and more popular, there are numerous researchers who are contributing to this field. Knowing others contribution can not only give us a better understanding of current progress towards this problem but also avoid producing duplicate work. After reviewing most of the research papers related to small object detection, we found that all the effective improvements are made via one of the three approaches:

- Oversample and Augment the small object samples
- Improve the training scheme by modifying the loss function
- Modify the existing model architecture or develop new innovations

## Related Work

Good detection models always require high quality training datasets in the first place. The reason is that the only reliable learning source for machine learning models is from the training data. If the model is lack of good training samples, it would be nearly impossible for it to succeed. Most of the public object detection datasets appear to have more large objects than small objects overall. For example, in MS COCO, only about half of the training images contain small objects, whereas more than 80% of the training samples contain large objects. (Lin et al. 2015) Therefore, lack of small training objects should be considered to be the first issue we need to tackle. It is found that the actual IoU between small ground-truth objects and the predicted anchors is a lot lower than the expected IoU threshold. The main reason of such outcome is that the model is lack of small objects samples during training, so oversampling and augmentation strategies become a promising solution in this case. Firstly, they

experimented with different oversampling ratios. It is observed that larger oversampling could increase the average precision of small objects slightly, while damaging the AP of medium/large size objects a little. It is implied that the ratio must be chosen based on the relative importance between small and large objects. In addition, they tried to copy and paste the small objects multiple times in random places. It is shown that a certain number of duplicates training samples can help boost the performance slightly. More specifically, small number of copy-pasting can help the model achieve better performance, but too many duplicates may lead to an overfitting problem. (Kisantal et al. 2019) Sometimes, if the model is supposed to be invariant to rotations, we should also augment the data with a number of randomized rotations. In this case, we may copy and paste the small objects with different rotations to achieve better results. Overall, their experiments proved that oversampling small object samples to some degree can easily improve the model performance. It is also proved that data augmentation can certainly help the model perform better on small objects, but there is a trade-off between the quality of predictions for small and large objects.

After having high-quality training datasets, the model is still not guaranteed to perform well. The next part we can probably improve is how we train the model. If the model is not trained properly and effectively, it might not behave as what we expected. It is often the case that some models can detect some certain classes more accurately than other classes. As we mentioned before, small objects are known to be harder to detect than large objects. According to Focal Loss for Dense Object Detection, Focusing training on a sparse set of hard examples and preventing the easy samples from overwhelming the detector during training can possibly overcome the class imbalance problem. To achieve this, they propose a new loss function that acts like a dynamically scaled cross entropy loss. More specifically, they propose to add a modulating factor to the cross entropy loss with a tunable focusing parameter. Intuitively, the modulating factor reduces the loss contribution from easy examples and penalizes harder on the hard examples.(Lin et al. 2018) In this way, the model can pay more attention to those difficult samples. If some classes have smaller number of traning instances, higher penalties can make the model learn a little more from the limited resources. In addition, if some classes are very hard to understand, the focal loss can help the model identify those problems and try to learn harder from them. Therefore, training with focal loss can make the model equally effective on all classes. As a result, small classes can be trained with higher attention, so the overall performance of the model will be improved.

Other than a good data set and a good training scheme, a good model is also crucial for small object detection. One of conventional method for small object detection and object detection in general is sliding window technique. This technique basically divides an image into thousands even millions of sub-windows and extract features and run object classification algorithms on those sub-windows (Fidler ). To detect small objects as well as big objects, the algorithm will first pre-process the image into an image pyramid

| | Segmentation AP | | | | Detection AP | | | |
|---|---|---|---|---|---|---|---|---|
| | small | medium | large | all | small | medium | large | all |
| baseline | 0.113 | 0.300 | 0.418 | 0.28 | 0.167 | 0.329 | 0.393 | 0.303 |
| oversampling 2× | 0.120 | 0.299 | 0.409 | 0.279 | 0.173 | 0.328 | 0.387 | 0.304 |
| oversampling 3× | **0.123** | 0.300 | 0.404 | 0.279 | **0.177** | 0.329 | 0.382 | **0.305** |
| oversampling 4× | 0.120 | 0.299 | 0.398 | 0.276 | 0.174 | 0.329 | 0.374 | 0.302 |

Figure 1: Comparison of AP by Oversampling with different ratios

| | Segmentation AP | | | | Detection | | | |
|---|---|---|---|---|---|---|---|---|
| | small | medium | large | all | small | medium | large | all |
| baseline | 0.113 | 0.300 | 0.418 | 0.280 | 0.167 | 0.329 | 0.393 | 0.303 |
| 1× pasted | 0.122 | 0.303 | 0.405 | 0.281 | 0.174 | 0.333 | 0.384 | 0.307 |
| 2× pasted | 0.121 | 0.300 | 0.406 | 0.279 | 0.175 | 0.329 | 0.385 | 0.305 |
| 3× pasted | **0.124** | 0.303 | 0.411 | **0.281** | **0.178** | 0.331 | 0.387 | **0.306** |
| 4× pasted | 0.120 | 0.299 | 0.403 | 0.279 | 0.179 | 0.329 | 0.382 | 0.305 |
| 5× pasted | 0.119 | 0.301 | 0.412 | 0.279 | 0.176 | 0.330 | 0.389 | 0.305 |

Figure 2: Comparison of AP by pasting different times

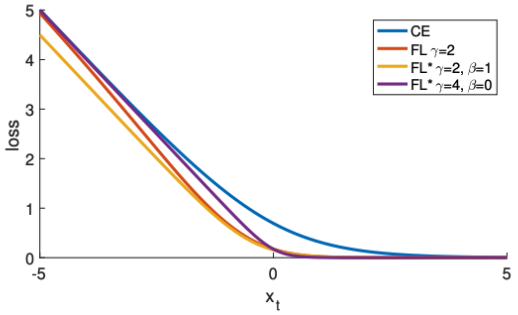


Figure 3: Focal Loss tuned with different parameters

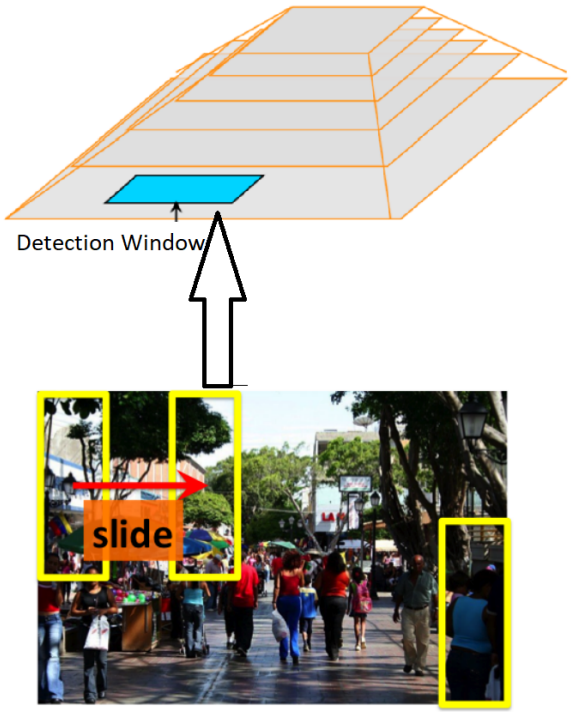


Figure 4: Illustrating how sliding window works with image pyramid

with different resolution (called scale). The windows slides over all levels of the pyramid (as illustrated by figure), and the ones at the bottom with higher resolutions will help finding the smaller objects and the ones at the top will help detecting bigger objects in the image. This method is slow and inefficient since it basically exhaustively searched the entire image pyramid. The limitation of the conventional classifier also makes it hard to detect multiple objects of different type as the conventional classifier is trained to classify one type of object only. In 2018, a new model is proposed which combines the idea of image pyramid with deep learning (Hu et al. 2018). Instead of building pyramids on the image directly, this model runs convolution operation on the image for five times, and take the output of the third, forth and fifth layer. The different scales of the image is built, not on the image, but rather on the convolutional layers, which is much easier to deal with compared to pixels. Each time the convolution operation is done on the previous layer, the output will contain more abstract semantic features and less detailed resolution. Then the model normalizes and concatenates the three output layers into a one-dimensional array for object classification purpose. This model yields detection accuracy for small object 11% higher than the best previous model. This method is also faster and more efficient compare to the sliding window model described above (Hu et al. 2018).

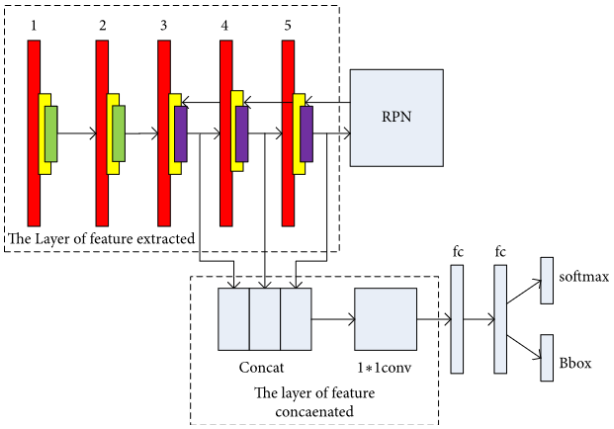


Figure 5: Illustrating how the multiscale detection network runs with convolutional network

Image pyramid is good, but the problem is large image pyramid is memory consuming and not efficient in general. Therefore, some researchers have decided to take the idea of multi-scaling and pyramid and built FPN (Feature Pyramid Network) model to further improve the performance on small object detection. This model, like the multi-scaling model, abandoned building pyramid by scaling the image, it scales features of the image. Under this model, the image goes through a bottom-up pathway to produce a feature pyramid using the feed-forward computation of the chosen convolutional neural network. By property of this calculation, the upper layer of the pyramid would have low resolution and semantically strong features while the lower layer will have high resolution and semantically weak features. The pyramid generated from the bottom-up pathway then go through a top-down pathway by up-sampling the higher layer and add it to the layer one level below it (which undergoes a 1x1 convolutional layer to match the channel dimension) as illustrated in figure 1. The bottom-up pathway merges the higher layer which is semantically stronger with lower layers which is more accurately localized. Object detection is then done on different layers of the feature pyramid after the top-down path according to the scale of the object. Larger scale object uses higher layer and smaller scale object uses lower layer. FPN nearly double the accuracy on small objects [detection] (Lin et al. 2017) compared to previous proposed methods for mask proposal generation. The reason behind this is FPN leverages contextual information (Lin et al. 2017) as the semantically stronger feature (the context) is combined with the higher resolution (the details) in the top-down pathway gives a more precise prediction of existence of small object. Further improvement and optimization on this idea has also been investigated to make the model faster and more accurate. PANet (Path Aggregation Network) is proposed as an augmentation to FPN. PANet improves on FPN by adding an extra top-down path so that the entire network would benefit from the accurate localization signals in lower layer (Liu et al. 2018). This bottom-up pathway "shortens the information path between lower layers and topmost feature" (Liu et al. 2018). It also

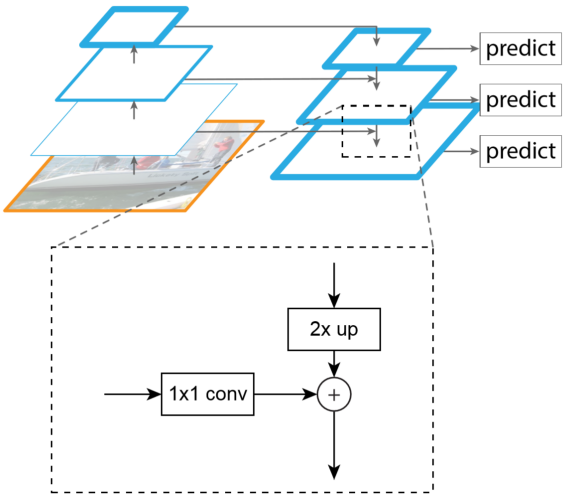


Figure 6: Illustrating the top-down pathway, merging higher layer with lower layer by addition

uses feature pooling to avoids inefficient assignment of layer and allowing small proposals to access these features better exploits useful context information for prediction (Liu et al. 2018) which improves the accuracy of small object detection. Another optimization is to use a better merging sequence between layers instead of the sequential merging in the regular FPN implementation. NAS-FPN is an implementation that uses this ideal. It takes advantage of the Neural Architecture Search and discovers "a new feature pyramid architecture in a novel scalable search space covering all cross-scale connections" that "consists of a combination of top-down and bottom-up connections to fuse features across scales" (Ghaisi et al. 2019). The constructed feature pyramid would perform better in object detection. The model "surpasses Mask R-CNN detection accuracy with less computation time" (Ghaisi et al. 2019). The discovered network also finds "the connection between high resolution input and output feature layers" (Ghaisi et al. 2019) which is important for small object detection.

## RCNN

Another popular methods to solve small object detection which is F-RCNN(Fast-RCNN). The fundamental of F-RCNN and RCNN is Convolutional Neuro Network. Convolutional Neural Network is now a common way to deal with image classification. Basically, we pass an image to the network, and it is then sent through various convolutions and pooling layers. Finally, we get the output in the form of the objects class. We can use this classification concept to detect objects in image. The object detection has several steps:

1. Take an image as input.
2. Divide the image into various regions.
3. Consider each region as a separate image.
4. Pass all these regions(image) to the CNN and classify them into various classes.
5. After dividing each region into its corresponding class, we combine all these regions to get the original image with the detected objects.

The problem for the initial method is that each objects ratio in the image is different. As a result, we have many regions which takes a lot computational time. Region based Convolutional Neural Network(RCNN) is derived to solve this problem. Instead of dividing regions in the image, RCNN use a bunch of boxes in the image to check if each object is contained in some of the boxes (also called region). In terms of regions identification, RCNN method uses four identifiers color, varying scales, colors and enclosure. The steps RCNN takes are as followed:

1. Take an image as input.
2. Use selective search or other proposal method to retain Regions of Interests (regions containing some object).
3. All these regions are reshaped as input of the CNN and each region is passed to CNN.
4. CNN extracts features for each region and then Support vector machines are used to divide these regions into classes. Abounding box regression is used to predict the bounding boxes for each identified region.
5. Abounding box regression is used to predict the bounding boxes for each identified region.

Since RCNN uses three models which are CNN for feature extraction, Linear Support Vector Machine classifier for identifying objects and Regression model for tightening the bounding boxes, RCNN is quite slow to make prediction for each input image (around 40-50s).

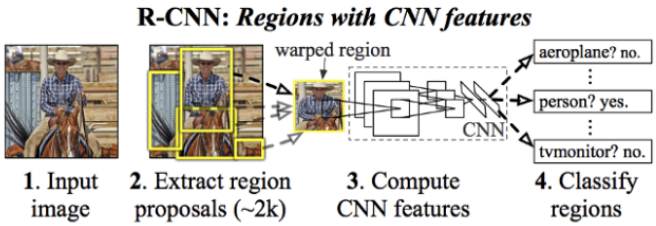


Figure 7: RCNN(Girshick and Microsoft Research 2015)

## FRCNN

Fast RCNN(F-RCNN) is developed to reduce the computational time that RCNN takes by reducing the times we run CNN in each image. In RCNN, lets say we have 2000 regions in each image then we need to run 2000 times CNN for each image. While in F-RCNN, we only need to run one time CNN for each image regardless of how many regions we divide. In F-RCNN, it only uses a single model, CNN, which extracts features from the regions, divides them into corresponding classes and returns the separated boundary boxes for each identified object class.(Ren, Zhu, and Xiao 2018) The steps of F-RCNN is as followed:

1. Take an input image.
2. Pass the image to a ConvNet and returns regions that containing objects (Regions of Interest).
3. Apply Regions of Interest pooling layer on the extracted Regions of Interest to ensure all the regions have the same size.
4. Pass the regions generated from the previous to a fully connected network, which would classify them and returns the boundary boxes by using Softmax and Linear Regression layers together.

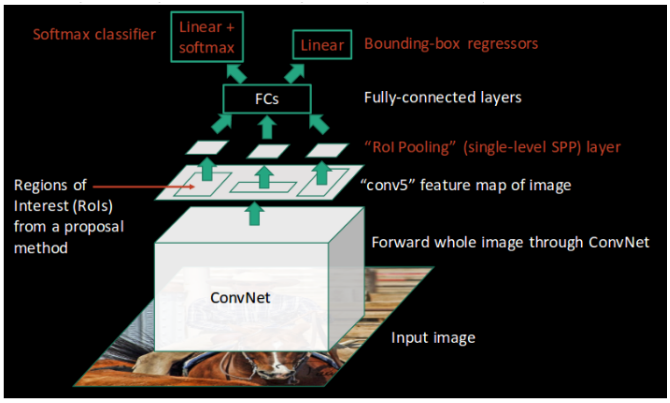


Figure 8: Fast-RCNN

The problem for the original F-RCNN in small objects detection is rich representation are difficult to learn from small objects poor-quality appearance and structure. Hence, F-RCNN is modified by two main factors, anchor size and super resolution, in order to increase the feature resolution of small objects and possibility for small objects to be detected. The common used RPN (region proposal network) anchor boxes need to be replaced in this scenario since they are much bigger than the most instances of our remote sensing object data set. According to Anchor Box Optimization for Object Detection, in terms of anchors, we adopt three scales with box areas of $16^2$, $40^2$, and $100^2$ pixels, and three aspect ratios of 1:1, 1:2, and 2:1(Zhong et al. 2018), which are adjusted for better coverage if the size distribution of our optical remote sensing data set. This paper also stated that appro-

| Method | mAP (%) | AP (%) | | Anchor Scale | Context | RR | BS |
|---|---|---|---|---|---|---|---|
| | | Ship | Plane | | | | |
| Baseline (stride = 16) | 66.6 | 58.2 | 75.0 | $\{128^2\ 256^2\ 512^2\}$ | | | |
| | 67.1 | 59.5 | 74.6 | $\{64^2\ 128^2\ 256^2\}$ | | | |
| | 68.1 | 60.1 | 76.0 | $\{10^2\ 40^2\ 100^2\}$ | | | |
| | 73.5 | 71.0 | 76.0 | $\{10^2\ 40^2\ 100^2\}$ | | | |
| Modified Faster R-CNN (stride = 8) | 74.1 | 71.7 | 76.5 | $\{10^2\ 40^2\ 100^2\}$ | √ | | |
| | 75.8 | 69.7 | 81.8 | $\{10^2\ 40^2\ 100^2\}$ | | √ | |
| | 76.7 | 69.8 | 83.6 | $\{10^2\ 40^2\ 100^2\}$ | | | √ |
| | 76.1 | 71.4 | 80.8 | $\{10^2\ 40^2\ 100^2\}$ | √ | √ | |
| | 77.1 | 70.4 | 83.9 | $\{10^2\ 40^2\ 100^2\}$ | | √ | √ |
| | 78.3 | 72.3 | 84.3 | $\{10^2\ 40^2\ 100^2\}$ | √ | | √ |
| | 78.9 | 72.9 | 85.0 | $\{10^2\ 40^2\ 100^2\}$ | √ | √ | √ |

Figure 9: The result of anchor size modified Fast- RCNN

priate anchor boxes can conduce to boosting the detection performance by almost two percentage small points, which evidently shows that adjusting the anchor size works significantly in small objects detection. Another way to adjust our F-RCNN model to small objects detections is through producing high-resolution feature maps. Super-resolution network is introduced to increase the detection resolution by up-sampling a small a small object image to a larger scale. Instead of resizing images, we can also enhance the image resolution and with a better resolution, the small object can be more easily to be detected. In 2014, Dong chao et al purposed "Learning a deep convolutional network for image super-resolution", where they developed a method called SRCNN(Super Resolution Convolutional Neural Network) and firstly introduce deep learning to solve Single Image Super Resolution, which had a significance improvement compared to previous example based method.

## SRCNN

According to 'Learning a Deep Convolutional Network for Image Super-Resolution', the process of SRCNN is as followed:(Chao Dong and Tang 2014)

1. Preprocessing: Upscale the image to the desired size using bicubic interpolation Y. (lower down the resolution)
2. Feed the image into the network
3. Convert the input to YCbCr color space (since only luminance channel (Y) is used by the network $conv1 \rightarrow relu1 \rightarrow conv2 + relu2 \rightarrow conv3$
4. Merge the output with interpolated CbCr channels to produce the final image.

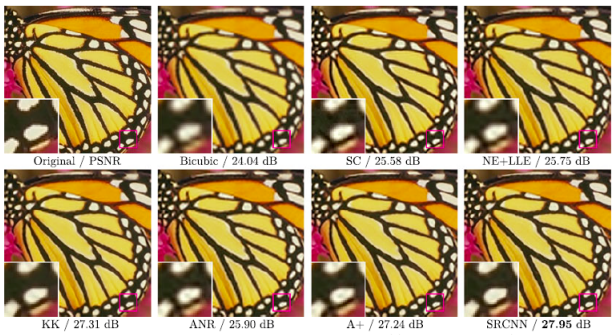


Figure 10: SRCNN results compared with example based methods

It outperforms the previous example based methods (figure 2) and it only consists of 3 convolutional layers which are:

1. Patch extraction and representation: this operation extracts (overlapping) patches from the low-resolution image Y and represents each patch as a high-dimensional vector. The dimension of the vector represents the set of feature maps of the image. The first layer is expressed as:

$$F_1(Y) = max(0, W_1 * Y + B_1)$$

Where:

$W_1$ = filters which has size $c * f_1 * f_1 * n_1$
$B_1$ = biases
c = the number of channels in the input image
$f_1$ = spatial size of a filter
$n_1$ = number of filters
$B_1$ = an $n_1$ dimensional vector, whose each element is associated with a filter.

2. Non-linear mapping: this operation maps the vector we get from the first convolutional layer to another high-dimensional vector which represents the high-resolution feature maps. . The second layer is expressed as:

$$F_2(Y) = max(0, W_2 * F_1(Y) + B_2)$$

Where:

$W_2$ = filters which has size $n_1 * 1 * 1 * n_2$
$B_2$ = an $n_2$ dimensional vector

3. Reconstruction: this operation aggregates the high resolution patch we get from the second convolutional layer and produce the final high resolution image. The output layer is expressed as:

$$F(Y) = W_3 * F_2(Y) + B_3$$

Where:

$W_3$ = filters which has size $n_2 * f_3 * f_3 * c$
$B_3$ = a c dimensional vector

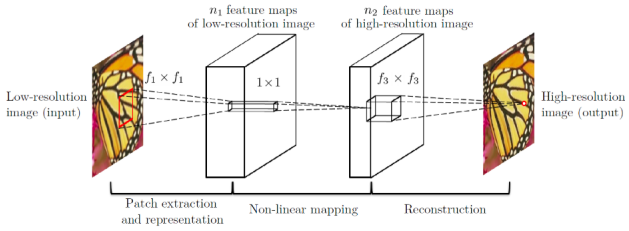


Figure 11: SRCNN visualization

SRCNN has only convolutional layers which provides the advantage that the input image can be of any sizes and algorithm is not patch-based. While the steps look simple, SRCNN is really difficult to train. Hyper parameter changes can greatly affect the SRCNN model and the parameters listed in the paper (learning rate = $10^{-4}$ for the first two layers, $10^{-5}$ for the last layer, SGD optimizer) leads PyTorch not to produce the optimal result.

**Loss function improvement** There are some modifications can be applied to enhance our SRCNN model. Mentioned in the paper, SRCNN used Mean Square Error(MSE) to train the model.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} |F(Y_i, \theta) - X_i|^2$$

As we see the result of the original model, the image we get by minimizing MSE is way too smooth since MSE tends to produce an image resembling the mean of all possible high-resolution pictures, resulting in a given low resolution picture. There is another problem of MSE MSE does not capture the perceptual differences between models output and the ground truth image. For example, We want to preserve the content of the original image, however, the resulting image contains several pixels of the original image shifted to the left. Thus, we replace Mean Square Error (MSE) in Per-pixel loss function method by Perceptual Loss function. While Per-pixel loss compare two images based on their individual pixel values, 'Perceptual Losses for Real-Time Style Transfer and Super-Resolution' stated that Perceptual Loss uses high-level image feature representations extracted from the pretrained CNN (Convolutional Neural Network).(Justin Johnson 2016) By replacing the loss function, we can upscale the result to be made up of objects resembling the original one as much as possible since network trained for image classification can store in its feature maps the information on what detailed of common objects look like.

While SRCNN is the first deep learning method to increase image resolution, there are quixte many new deep learning methods were developed and have much more improvement. According to SOD-MTGAN: Small Object Detection via Multi-Task Generative Adversarial Network, 'SRN can generate images of higher quality and less artifacts at large up-scaling factors (4x in our current implementation).'(Bai et al. 2018) Multi-task generative adversarial network (MTGAN) is really helpful in producing super-resolved images. Besides, it can distinguish the real high-resolution images from fake ones, predict object categories, and refine bounding boxes.

## Conclusion

In conclusion, there are overall three approaches to overcome small object detection problems. Firstly, small objects datasets should be oversampled and augmented in order to provide enough training data. In addition, certain improvements can also be made in the training scheme, such as modifying the loss function. Finally, developing new model architectures are quite necessary because most of the existing models have bottlenecks detecting small objects. In particular, we introduced the architectures of Feature Pyramid Network and Super Resolution which significantly improve the overall performance of the existing object detection models. As there are so many existing experiments done by other researchers, summarizing their work give us an overview of what kind of ideas and designs have been discovered by others. This article would be very meaningful for anybody who want to continue researching on the small object detection area. We typically gather the most popular and successful work in public, so this article would be a great start for any future usage. As we found in this article, there are a handful of new technologies that can be used in detecting small objects and they are showing some really promising evaluation results. In other words, detecting small objects is definitely not a dead end, and there are so many possibilities and potential to improve the overall performance further. Although new technologies improve the overall performance, the pre-

cision is still far from satisfaction. In the future, researchers can try to leverage multiple different existing methods to see how they interact. Sometimes good combinations of methodologies may give some surprising results. Moreover, we should always try to improve the quantity and quality of the training samples, especially for small objects. As we mentioned in this article, most of the existing public datasets did not collect enough instances for small objects. Last but not least, new innovations are always encouraged since the existing model architecture would never be perfect.

## References


Bai, Y.; Zhang, Y.; Ding, M.; and Ghanem, B. 2018. Sodmtgan: Small object detection via multi-task generative adversarial network. In *The European Conference on Computer Vision (ECCV)*.

Chao Dong, Chen Change Loy, K. H., and Tang, X. 2014. Learning a deep convolutional network for image super-resolution. *ECCV*.

Fidler, S. Lecture notes in intro to image understanding.

Ghaisi, G.; Lin, T.-Y.; Pang, R.; Le, V. Q.; and Google Brain. 2019. Nas-fpn:learning scalable feature pyramid architecture for object detection. *arXiv preprint arXiv:1904.07392*.

Girshick, R., and Microsoft Research. 2015. Fast r-cnn. *arXiv preprint arXiv:1504.08083*.

Hu, G. X.; Yang, Z.; Hu, L.; Huang, L.; and Han, J. M. 2018. Small object detection with multiscale features. *International Journal of Digital Multimedia Broadcasting* 2018.

Justin Johnson, Alexandre Alahi, L. F.-F. 2016. Perceptual losses for real-time style transfer and super-resolution. *arXiv:1603.08155*.

Kisantal, M.; Wojna, Z.; Murawski, J.; Naruniec, J.; and Cho, K. 2019. Augmentation for small object detection. *arXiv preprint arXiv:1902.07296*.

Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; and Girshick, R. 2015. Microsoft coco: Common objects in context. *arXiv preprint arXiv:1405.0312*.

Lin, T.-Y.; Dollr, P.; Girshick, R.; He, K.; Hariharan, B.; and Belongie, S. 2017. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*.

Lin, T.-Y.; Goyal, P.; Girshick, R.; He, K.; and Dollr, P. 2018. Focal loss for dense object detection. *arXiv preprint arXiv:1708.02002*.

Liu, S.; Qi, L.; Qin, H.; Shi, J.; Jia, J.; The Chinese University of Hong Kong; Peking University; SeseTime Research; and YouTu Lab Tencent. 2018. Path aggregation network for instance segmentation. *arXiv preprint arXiv:1803.01534*.

Ren, Y.; Zhu, C.; and Xiao, S. 2018. Small object detection in optical remote sensing images via modified faster r-cnn. *Applied Sciences* 8:813.

Zhong, Y.; Wang, J.; Peng, J.; and Zhang, L. 2018. Anchor box optimization for object detection. *arXiv preprint arXiv:1812.00469*.