Improving performance in object detection for small objects

Gijung Lee
Electrical and Computer Engineering department
University of Florida
lee.gijung@ufl.edu

***Abstract*—The detection of the small object has been challenging because of the limitation of the property of the convolutional neural network. For this project, I tried two methods to improve the performance of small objects detection. The first method is upscaling or improving the details of the image by using the concept of super-resolution. The second method is the process of performing prediction over small slices of the original image and then merging predictions from all sliced images on the original image. To detect small objects, I used the DOTA dataset [12] which contains aerial images. For the super-resolution method, I used the Efficient Sub-pixel Convolutional Neural Network (ESPCN). With these methods, I could get 27.5% mAP, 46.4% mAP_50, 28.0% mAP_75 from 15.7% mAP, 26.3% mAP_50, 15.6% mAP_75 without any methods.**

***Index Terms*—Object detection, Super-resolution, Slicing images**

I. Introduction

Overall, the object detection problem has been researched a lot after the development of the Convolution neural network. While the performance on medium and large-size objects has been increased, the detection of small objects has been challenging, especially in remote sensing images. The low resolution and simple shape of most of the small objects caused poor performance on small object detection. For small objects in remote sensing images, just a few pixels represent the whole object. With a few pixels that represent the whole object, it is difficult to identify and detect the object.

II. Description

A. Convolution neural network

Most of the object detection algorithms are based on Convolution neural networks (CNNs). The CNNs can create low-level abstractions of the image. Then, the low-level abstractions such as lines and circles are combined to detect objects that we want. This is a very powerful property to detect the object. However, this can also be a problem to detect small objects. As you can see the figure 1, this network has several convolution layers and pooling layers that affect to reduce the resolution of the image. This makes the resolution of the original image down to ~30 x 30 resolution. [10] The features of the small object that are extracted on the first layer easily disappear passing the remained layers and cannot reach detection and classification steps. Due to this property of the CNNs, small object detection has been challenging than other medium and large objects.

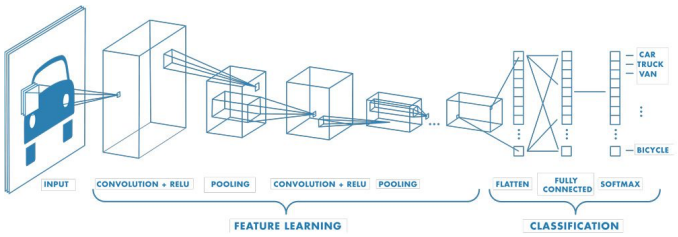

Fig. 1. Structure of the CNNs. [1]

The main problems of small object detection are as follows:

- A small object has a few pixels that represent the whole object.
- The features of the small object extracted on the first layers disappear passing the remained layers.

As I could check the problems of the small object detection, I realized that it is very important to upscale the resolution of the image making the features of the small object to be extracted more.

For these problems, I used two methods. First, I tried to upscale the resolution of the image. Simple upscaling is a way that stretches the low-resolution image onto the larger image. To be a higher resolution image, the low-resolution image is interpolated by copying and repeating pixels in neighbors. However, normally, this method doesn't have good results. To improve the resolution of the image and get better results, I tried the super-resolution method instead of this simple upscaling. For this super-resolution method, I used the Efficient Sub-pixel Convolution Neural Network (ESPCN). Second, I tried to get predictions over sliced images from the original image and then merge the predictions on the original image.

III. Related Work

A. Super-Resolution

Single image super-resolution is an important method in image processing to recover a high-resolution image from its corresponding low-resolution image. There are several super-resolution models based on deep neural networks. For instance, Super-Resolution Convolutional Neural Network (SRCNN) [4], Fast Super-Resolution Convolutional Neural Network FSRCNN) [5], the efficient sub-pixel convolutional neural network (ESPCN). The SRCNN [4] and FSRCNN [5] based on the convolutional neural network have some weaknesses. These CNN approaches must use interpolation methods such as bicubic interpolation. Moreover, these approaches use the up-sampled low-resolution image to apply the convolutional neural network. This method causes increasing computational complexity and memory cost. [2] To avoid these problems, I used the ESPCN method that adds an efficient sub-pixel convolutional layer to the CNNs.

B. An Efficient Sub-Pixel Convolutional Neural Network

The Efficient Sub-Pixel Convolutional Neural Network (ESPCN) upscales the resolution at the end of the network. [2] In ESPCN, the upscaling step is applied in the last layer. For this reason, the low-resolution image can be fed to the network unlike the SRCNN [4] and FSRCNN [5]. With this benefit, the ESPCN does not need to use the interpolation method. Moreover, the reduced input image size can make the model use a smaller kernel size that is used to extract features. I can get better efficiency by the reduced computational complexity and memory cost. [2]

1) Network Structure

ESPCN has several convolutional layers that obtain feature maps of the input images and an efficient sub-pixel convolutional layer that recovers the output image. Usually, the network has three convolutional layers and a sub-pixel convolution layer.

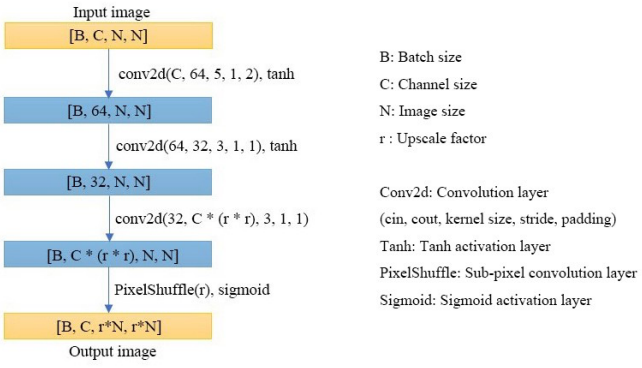

Fig. 2. ESPCN model. [3]

The image passes the first layer that has a convolutional layer with 64 filters and the kernel size of 5 × 5, followed by a tanh activation layer. Then it passes the second layer that has a convolutional layer with 32 filters and the kernel size of 3 × 3, followed by a tanh activation layer. Lastly, it passes the third layer that has a convolutional layer with the fixed number of output channel C × r × r and the kernel size of 3 × 3. It is applied the sub-pixel shuffle function to make the output image will have the shape [B, C, r × N, r × N], followed by a sigmoid activation layer. [3] I may say that the super-resolution is an estimation of how similar a high-resolution image (HR) I^{SR} given by a low-resolution image (LR) I^{LR} that is downscaled from the original Image I^{HR} and the original Image I^{HR}. The I^{LR} and I^{HR} have C channels. Respectively, they can be represented as $H \times W \times C$ and $rH \times rW \times C$. [2]

2) Sub-pixel Convolution

Convolution with a stride of $\frac{1}{r}$ in the low-resolution space with a filter W_s of size k_s with weight spacing $\frac{1}{r}$ can activate different parts of W_s for the convolution. [2]

$$I^{SR} = f^L(I^{LR}) = \mathcal{PS}(W_L * f^{L-1}(I^{LR}) + b_L)$$

The periodic shuffling operator ($\mathcal{PS}$) make the $H \times W \times C \cdot r^2$ tensor to a tensor of shape $rH \times rW \times C$.

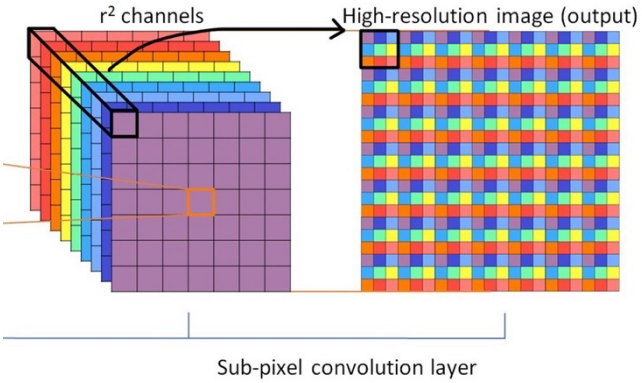

Fig. 3. Operation of pixel shuffling. [3]

As you can see in figure 3, $r \times r$ square feature maps on multiple channels are combined into a single channel in a high-resolution image. Thus, each pixel on feature maps can be equivalent to the sub-pixel on the generated output image. [3] The interpolation method is indirectly included in the convolutional layers so we can use a low-resolution image on the first convolution layer unlike other SRCNN [4] and FSRCNN [5].

3) Loss Function

To measure the difference between the SR images and the HR original images, the mean squared error (MSE) is used.

$$\ell(W_{1:L}, b_{1:L}) = \frac{1}{r^2 HW} \sum_{x=1}^{rH} \sum_{x=1}^{rW} \left(I_{x,y}^{HR} - f_{x,y}^{L}(I^{LR}) \right)^2$$

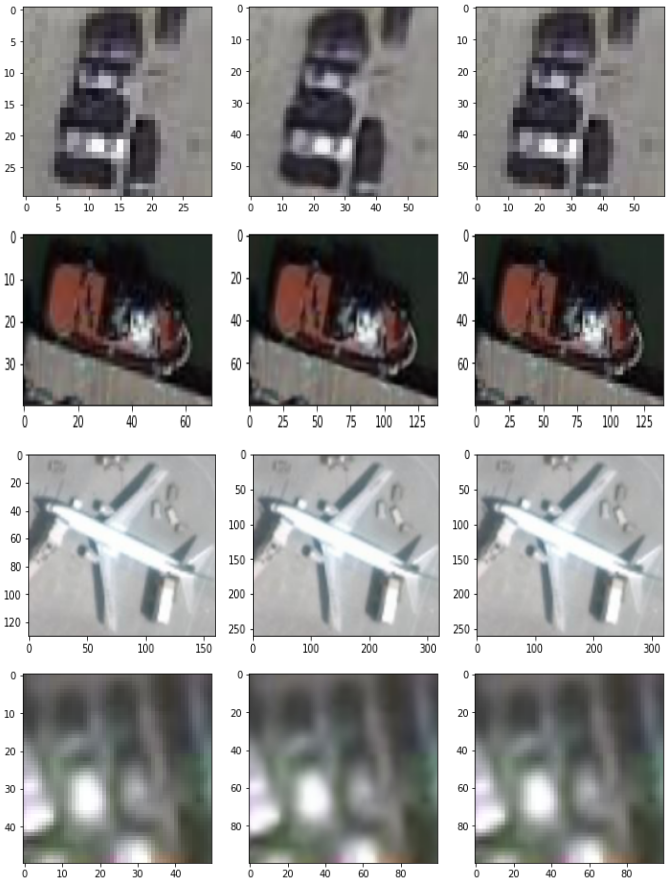

Fig. 4. Super-resolution examples, from left to right: Original, ESPCN, and Pixel replication. From top to bottom: small vehicle, ship, plane, and small vehicle.

C. *Faster RCNN*

Faster RCNN [6] has two parts. The first part is a deep convolutional network that proposes regions. The second part is a detector that uses the proposed regions. The region proposal network (RPN) acts as the 'attention' of this network. [6]

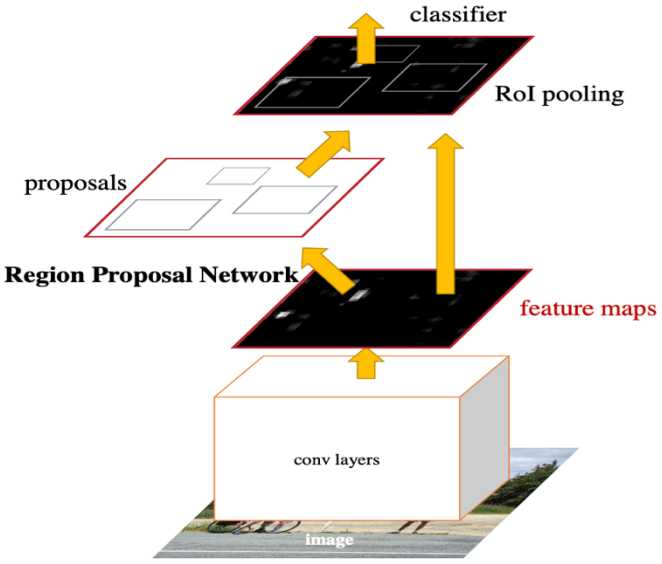

Fig. 5. Faster RCNN. [6]

1) *Region Proposal Networks*

A region Proposal Network (RPN) is a fully convolutional network that creates proposals with various scales and aspect ratios. [7]

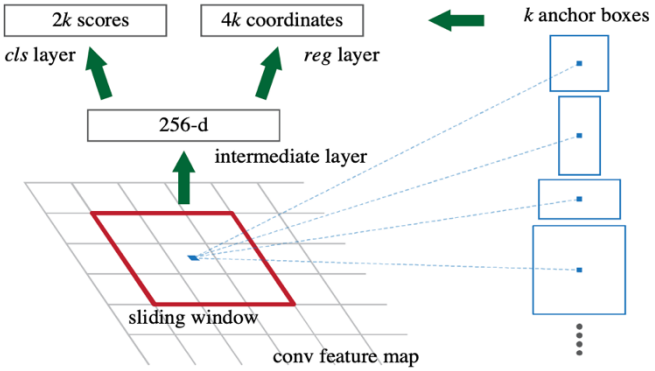

Fig. 6. Region Proposal Network (RPN). [6]

2) *Anchor*

As you see in figure 6, the feature map from the last convolution layer is sliced by a rectangular window of size n × n. k region proposals are generated each window. Each region proposal is proposed according to an anchor box that has a scale and aspect ratio. With these anchor boxes, the model can be offered a scale-invariant object detector as a single image at a single scale is used. This is useful to avoid using multiple images or filters.

D. *Fine Tuning*

Fine-tuning takes a trained model for a given task and makes the model perform a similar task. [11] Without training the model from scratch to get a feature extraction, using a pre-trained model can give us the feature extraction that happens in the top layer of the model. With the fine-tuning method, I can save time and get better results because the pre-trained model allows me to have a model that is trained with more data.

E. *Sliced inference*

I tried to make the patches on the images for the prediction part. I assumed that detecting in a small area on the images helps to detect objects better. This idea was come out in my experience. I usually see the small area to check small objects with a hand magnifier. I could find code for this method: https://github.com/obss/sahi/blob/main/demo/slicing.ipynb

IV. Evaluation

To evaluate the performance of two methods. I tried 3 types of experiments. Originally, I planned to make an end-to-end model that includes upscaling images and sliced inference. However, I could not approach this trial because of the limits of the memory and the computational constraints. I changed the trial making two parts of the process. First, I had a process that makes upscaled images. Second, I had a process that get predictions with images from the first process. With this process I could tried experiments without any problems.

A. Data Set

For this project, I used the DOTA data set [12] version1.0. DOTA includes large-scale aerial images for object detection. To get the images, different sensors and platforms are used. The size of an image in DOTA data is in the range of 800×800 to 20,000 × 20,000 pixels. DOTA-v1.0 contains 15 common categories, 2,806 images and 188, 282 instances. The proportions of the training set, validation set, and testing set in DOTA-v1.0 are 1/2, 1/6, and 1/3, respectively. [12]

For these experiments, I used the pre-trained Faster-RCNN [6] model. However, it has a problem to do fine-tuning the model with the DOTA dataset [12]. There were two main reasons for this problem. First, the pre-trained model is trained on medium and large-scale objects. Second, the input images should be rescaled to small size, e.g. (1333 × 800). Rescaling the size of an image causes loss of the information that the image has. Moreover, it causes that the small objects in the image cannot be visible and the model cannot have a chance to detect the small objects. To solve this problem, I used random crop in the data preprocessing part. This random crop method can make the model get images without losing the small objects.

B. Comparing results of training between using the upscaled images and original images

First, I trained models with two conditions to check the effects of the upscaling of images. The first one is to train with original images that are not applied super-resolution method. The second one is to train with upscaled images that are applied super-resolution method. As you see in table 1, in two conditions, the model trained with upscaled images has better results. However, this difference is not big enough.

1) Training with the original image – 1X

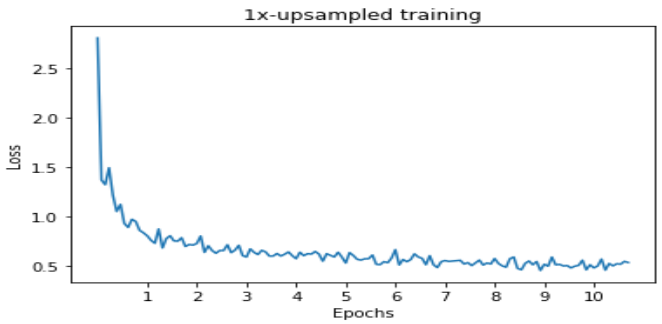

Fig. 7. Loss-curve on the original images.

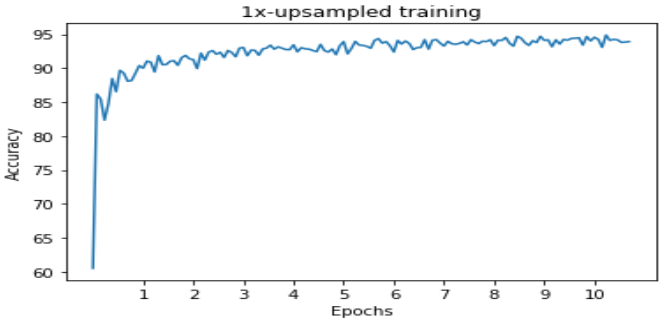

Fig. 8. Accuracy on the original images.

1) Training with upscaled images – 2X

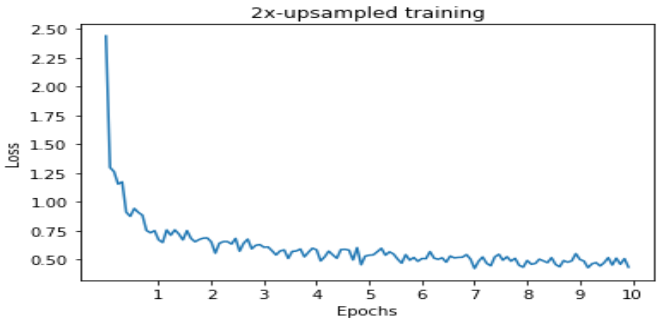

Fig. 9. Loss-curve on the upscaled images – 2X.

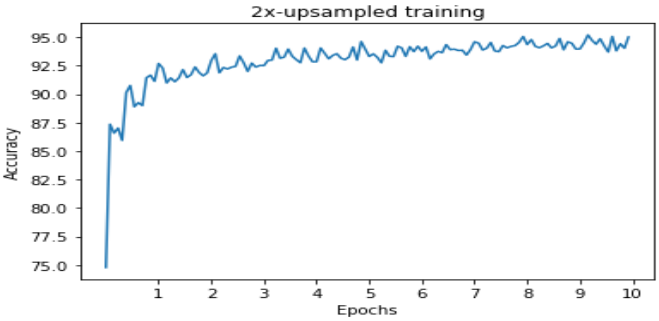

Fig. 10. Accuracy on the upscaled images – 2X.

Train	Loss	Accuracy
1x-Image	0.45222	94.87%
2x-Image	0.42427	95.19%

Table. 1. Loss and Accuracy on DOTA-v1.0 train set with Fast RCNN

As you can see the table 2, I got unexpected results of mean average precision (mAP). Even the model that trained with upscaled images has better accuracy on the train set, the actual results on the validation set have worse mean average precision. With this experiment, I could not check the effect of the upscaled images in object detection.

Validation	mAP	mAP_50	mAP_75	mAP_S	mAP_M	mAP_L
1x-Image	15.7%	26.3%	15.6%	2.8%	12.4%	30.1%
2x-Image	10.9%	18.3%	10.8%	0.1%	3.9%	14.4%

Table. 2. Results on DOTA-v1.0 validation set with Fast RCNN. – without sliced images inference.

Leaving these results behind, I tried another experiment. In this experiment, I tried sliced inference. In this case, I could check the effect of prediction by patched images. As you see the table 3, in both conditions, the results have shown improvements, especially, the model trained with upscaled images has improved more than 100%. Furthermore, I tried to see actual predictions on the test set and compare them with the effects of those methods. In random test images, I could find the effect of the sliced inference. As you see in figures 11 and 12, You can find a big difference between the predictions without sliced

inference and with sliced inference.

Validation	mAP	mAP_50	mAP_75	mAP_S	mAP_M	mAP_L
1x-Image	26.9%	45.4%	27.1%	8.2%	27.1%	39.8%
2x-Image	27.5%	46.4%	28.0%	3.4%	21.7%	32.2%

Table. 3. Results on DOTA-v1.0 validation set with Fast RCNN. – with sliced images inference.

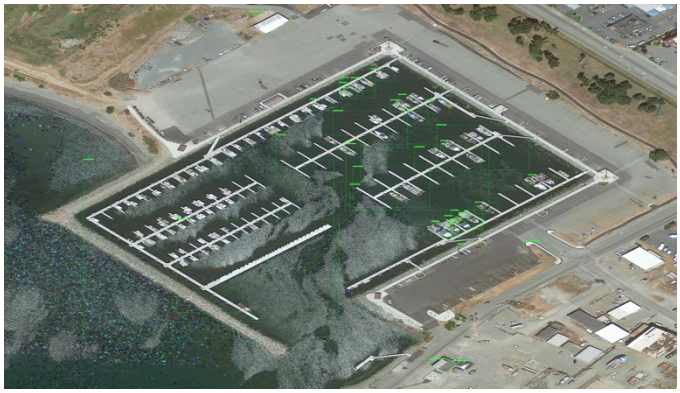

Fig. 11. Prediction on the test image without sliced inference.

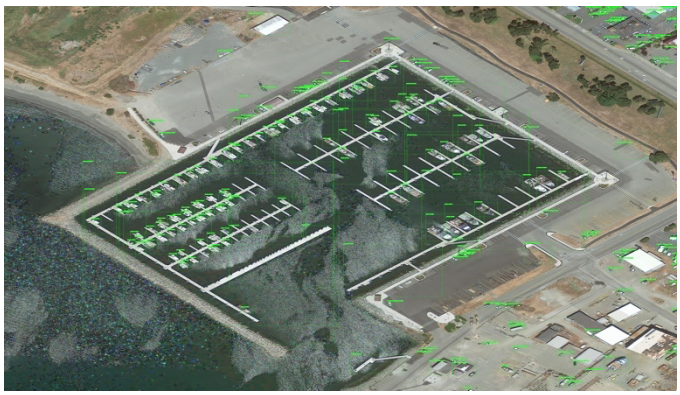

Fig. 12. Prediction on the test image with sliced inference.

Also, I tried to find the effect of upscaling the image by super-resolution. In this trial, I tried two kinds of images. First, I check the result on a low-resolution image. As you see in figures 13 and 14, super-resolution (ESPCN) helps to detect small objects.

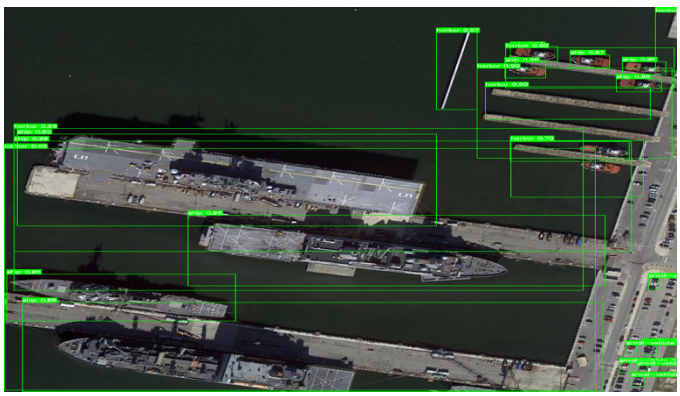

Fig. 13. Prediction on the low-resolution test image with sliced inference. – without upscaled by super-resolution.

Fig. 14. Prediction on the low-resolution test image with sliced inference. – with upscaled by super-resolution.

Also, as you can see in figure 15, I could get better results with high-resolution images

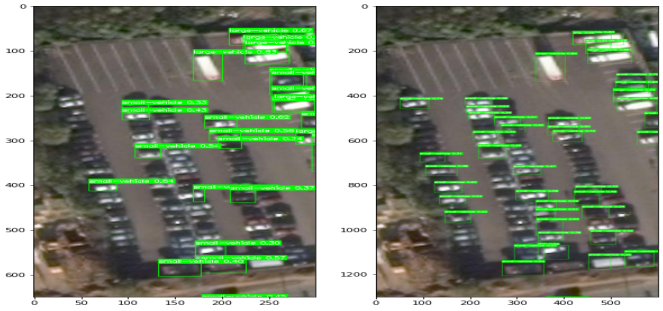

Fig. 15. Prediction on the high-resolution test image with sliced inference. – Left: without super-resolution, Right: with super-resolution. These images are cropped to show the difference. You can check the full image in Supplemental Material

Finally, I wanted to check the difference between two models that trained without super-resolution and with super-resolution respectively and detect with the upscaled image.

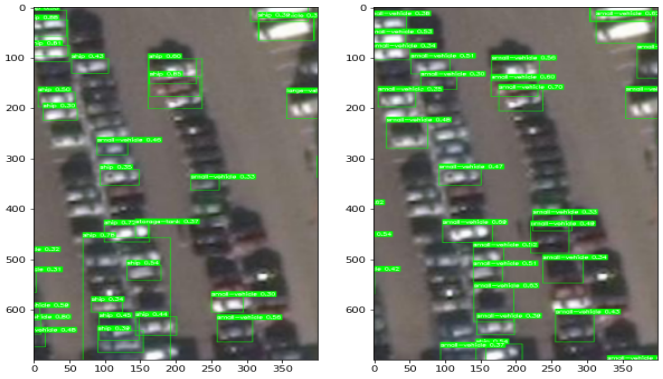

Fig. 16. Prediction on the high-resolution test image with sliced inference. – Left: trained without super-resolution but detect with the upscaled image, Right: with super-resolution. These images are cropped to show the difference. You can check the full image in Supplemental Material

.

As you see the figure 6, both models predict small objects well, however, the left image that is from trained without super-resolution shows bad classification on objects. This model

classifies the objects as a ship. Unlike the left image, the right image shows proper classification of the objects. This model classifies the objects as the small vehicle.

With these experiments, I could check the effect of the super-resolution method and the sliced inference method.

V. Conclusion

In this project, I tried to improve the performance of small object detection with two methods. With these methods, I could enhance the performance of the object detection more than 100% especially on the small objects. When the models predict the objects without sliced inference, the model trained with images that are not applied super-resolution got better results. However, if model used sliced inference, the model trained with upscaled images by super-resolution could get better results. I may need to research more about this result. The results show that using both methods may get a synergy effect. I may need to change the anchor size and the IoU scale to get better results when the trained model is evaluated. For better improvements on the object detections, I may need to try getting more data set by doing data augmentation. To get better results, more data set is a key point. My future work will focus on applying this method in the real world.

References

[1] *Introduction to deep learning: What are Convolutional Neural Networks? video*. Video - MATLAB. (n.d.). Retrieved December 14, 2021, from https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html.

[2] Shi, W., Caballero, J., Huszar, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., & Wang, Z. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[3] Cen, zhuo. (2020, April 17). *An overview of ESPCN: An efficient sub-pixel convolutional neural network*. Medium. Retrieved December 14, 2021, from https://medium.com/@zhuocen93/an-overview-of-espcn-an-efficient-sub-pixel-convolutional-neural-network-b76d0a6c875e.

[4] C. Dong, C. C. Loy, K. He, and X. Tang. Image super-resolution using deep convolutional networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2015.

[5] C. Dong, C. C. Loy, and X. Tang. Accelerating the super-resolution convolutional neural network. In *European Conference on Computer Vision (ECCV)*, 2016.

[6] Ren, S., He, K., Girshick, R., & Sun, J. Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[7] Gad, A. F. *Faster R-CNN explained for Object Detection Tasks*. Paperspace Blog. 2021, April 9. https://blog.paperspace.com/faster-r-cnn-explained-object-detection/.

[8] Krishna, H., & Jawahar, C. V. Improving small object detection. *2017 4th IAPR Asian Conference on Pattern Recognition (ACPR)*.

[9] Bashir, S. M., & Wang, Y. Small object detection in remote sensing images with residual feature aggregation-based super-resolution and Object Detector Network. *Remote Sensing*, *13*(9), 1854. 2021.

[10] Quantum. (2019, February 12). *Small objects detection problem*. Medium. Retrieved December 14, 2021, from https://medium.datadriveninvestor.com/small-objects-detection-problem-c5b430996162

[11] Elhamraoui, Z. *Fine-tuning in deep learning*. Medium 2020, June 29, from https://ai.plainenglish.io/fine-tuning-in-deep-learning-909666d4c151

[12] Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., & Zhang, L. Dota: A large-scale dataset for object detection in aerial images. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Supplemental Material

Fig. 17. Prediction on the high-resolution test image with sliced inference. – without super-resolution – image size: 7582×4333

Fig. 18. Prediction on the high-resolution test image with sliced inference. – with super-resolution – image size: 15164×8666

Fig. 19. Prediction on the high-resolution test image with sliced inference. – trained without super-resolution but detect with the upscaled image – image size: 15164×8666

Fig. 20. Prediction on the high-resolution test image with sliced inference. – trained with super-resolution but detect with the upscaled image – image size: 15164×8666