Object Detection in Aerial Images: What Improves the Accuracy?

Hashmat Shadab Malik
hashmat.malik@mbzuai.ac.ae

Ikboljon Sobirov
ikboljon.sobirov@mbzuai.ac.ae

Abdelrahman Mohamed
abdelrahman.mohamed@mbzuai.ac.ae

Abstract

Object detection is a challenging and popular computer vision problem. The problem is even more challenging in aerial images due to significant variation in scale and viewpoint in a diverse set of object categories. Recently, deep learning-based object detection approaches have been actively explored for the problem of object detection in aerial images. In this work, we investigate the impact of Faster R-CNN for aerial object detection and explore numerous strategies to improve its performance for aerial images. We conduct extensive experiments on the challenging iSAID dataset. The resulting adapted Faster R-CNN obtains a significant mAP gain of 4.96% over its vanilla baseline counterpart on the iSAID validation set, demonstrating the impact of different strategies investigated in this work.

1. Introduction

Object detection (OD), one of the computer vision tasks, poses its own challenges on top of mere identification or localization tasks [33]. It is a task of classification and localization of objects in an image. It has gained sufficient fame to be a major field of research in computer vision on account of its efficacy and wide use in real life applications. To make the task even more challenging, OD in aerial images is emerging as a new task in which minute objects are generally of interest of detection.

Traditionally, machine learning was the weak solution to OD tasks. An ensemble of hand-crafted feature extractors, such as histogram of gradients [5], were commonly in use. Recent strides in deep learning (DL) have proved its viability in OD tasks, yielding promising and applicable results. Not only did their advantage of automating the feature extraction process make the network construction effortless, they also produced a substantial boost in results. Therefore, the primary approach to tackling the OD tasks at present is DL-driven models.

In this work, we apply one of the most popular architectures for OD, Faster R-CNN [25], and make numerous modifications on top to reach higher results. Our contributions are as follows:

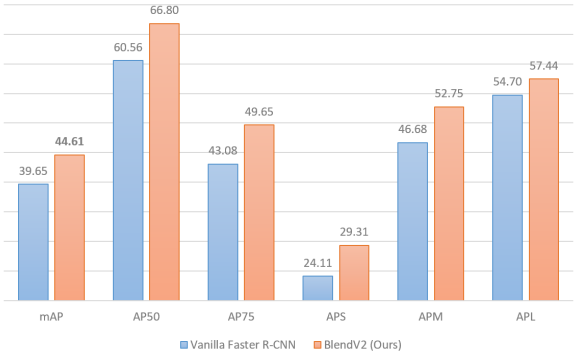


Figure 1: The figure shows the results of the improvements introduced on top of the vanilla Faster R-CNN.

- Examining a new backbone to the Faster R-CNN network
- Exploring several data augmentations that contribute to an increase in results
- Proposing a deeper region proposal network that integrates a spatial and channel squeeze and excitation block
- Investigating soft non-max suppression technique in the Faster R-CNN network

2. Related Work

Object detection task of computer vision on its own is challenging, and therefore, is attractive to many researchers in the field. It has been extensively studied with the use of natural images with promising models and results. Aerial images is newly emerging as an area of interest. It poses a few more issues on top of traditional OD, such as small scales and size irregularities in images. This section first highlights the common architectures used for OD, and second, reviews recent papers that tackled the OD task with natural as well as aerial images.

2.1. Architecture Review

Starting in early years of OD, traditional machine learning approaches were implemented to extract features; advances in deep learning has overthrown the traditional meth-

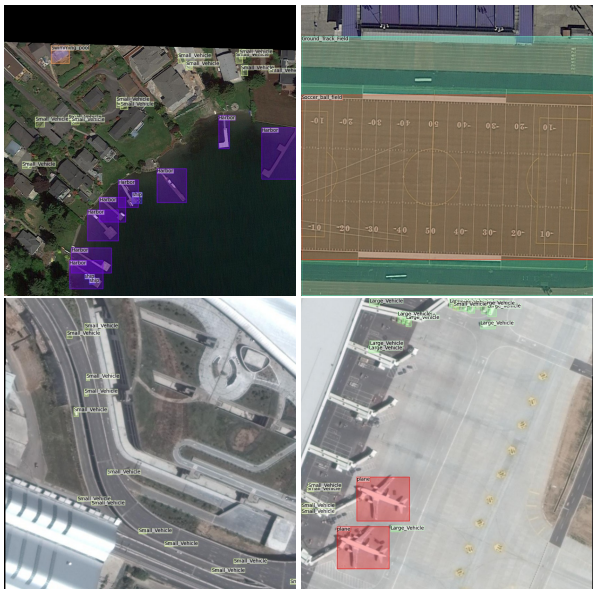


Figure 2: The figure shows a few samples from the iSAID dataset. Note that some images contain only one category, while others contain several within the same image.

ods, offering multiple advantages, including higher accuracy and precision. At an eagle view on current approaches, two categories can be extracted: one-stage OD and two-stage OD. Both categories are heavily in use today, with one-stage being much faster in inference time and two-stage yielding higher results. Of the many, YOLO family [22][23][24][1][35] is one of the most go-to methods in object detection for their continuous improvements in each newer version, both in time and accuracy. R-CNN family [10][9][25] comes toe-to-toe with the YOLO counterpart, with its higher results that is superior to YOLO. To be more specific, Faster R-CNN [25] is the last version of R-CNN. It is a single and unified network for object detection, and one of the most popular two stage detectors. It builds upon its predecessor Fast R-CNN [9] by introducing two new concepts: anchor boxes and region proposal network (RPN).

2.2. Recent Approaches

Building on top of the common architectures that yield promising results, multiple papers tried to make improvements in accuracy and/or time. In [31], the authors improve upon the vanilla SSD [18] to include multi-scale context information to reach higher accuracy. They use dilated and deconvolution layers in the context layers, and claim that they are the two variants of CSSD - a shorthand for context-aware single-shot multibox object detector. Further improvements on SSD were introduced in [19][20]. D2Det method presented in [3] is a two stage approach for object detection and instance segmentation in general. They validate their approach on MS COCO [17], UAVDT [7] and

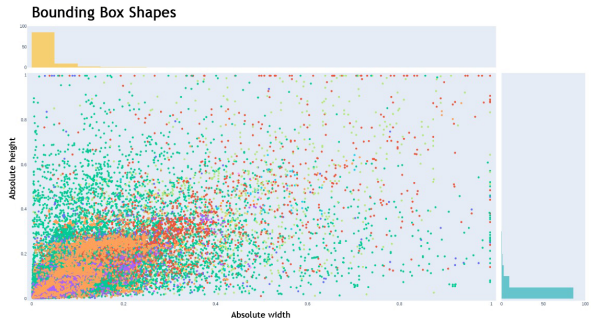


Figure 3: The figure shows height-to-weight correlation of bounding boxes of objects in the dataset. Each color represents a different category. Side bars are the proportion of how many instances of bounding boxes lie within that size range.

iSAID [29] for object detection and instance segmentation, outperforming other state-of-the-art methods at the time.

Several approaches have been proposed for object detection with aerial images. Igor et al. [36] present a simple CNN-based architecture to automate image classification and OD in aerial images. Sommer et al. [27] implement Fast R-CNN and Faster R-CNN to detect vehicles in aerial images, and validate their results on two publicly available datasets. In [32], ClusDet algorithm is proposed to detect objects in the same domain. To address the issues of small objects in aerial images and the non-uniformity in data, they incorporate a cluster proposal network to extract cluster regions, a scale estimation network and a separate detection network. To make the task even more challenging, [34] tackle the rotated OD (i.e. the bounding boxes are not axes aligned). They propose a dense anchor-free rotated object detector to do this task, with five parameters of prediction.

3. Datasets

Natural scene datasets as ImageNet [6], Pascal VOC [8], CityScapes [4] and MSCOCO [17] are the largest scale datasets available for use in deep learning (DL). However, their use is limited to upward orientation of the images, meaning that they are generally captured from a side view. When the viewpoint changes, for example, to a top view, the generalizability of the DL methods starts to deteriorate. In aerial images, the challenges to traditional imaging accumulate with a few more such as irregularities in shapes and orientations, object instances in higher densities, or large aspect and scale variations [29]. There are tenth of aerial, satellite and Google Earth datasets slowly emerging since 2008 up to now, and is still under much attention as a research area. Although the prevalence of such datasets is high, the total number of categories in each is usually limited to only a few. The datasets with the most number of

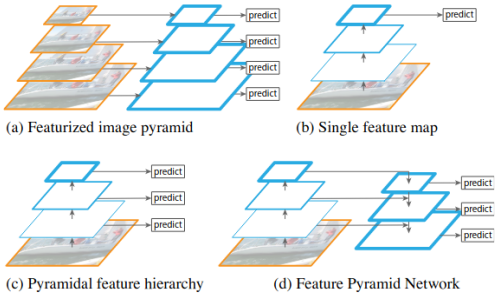


Figure 4: The figure shows different methods (a, c, d) to extract multi-scale information using different scales of feature maps, while also showcasing the simple single feature map (b) used in vanilla Faster R-CNN (Adapted from [16]).

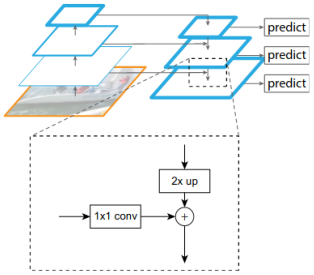


Figure 5: The figure shows the addition block showcasing how multi-scale feature maps are added in the FPN (Adapted from [16]).

categories and instances of these categories and the most popular ones are DIOR [15], DOTA [30], xVIEW [14] and iSAID [29], and iSAID dateset is used for the current work.

Unlike other counterparts, iSAID dataset is the first benchmark in instance segmentation task of computer vision using aerial images. 2806 high-resolution images that includes 655451 object instances for 15 categories are provided in the dataset. The categories are ship, storage tank, baseball diamond, tennis court, basketball court, ground track field, bridge, large vehicle, small vehicle, helicopter, swimming pool, roundabout, soccer ball field, plane and harbour. iSAID dataset images are of large resolutions and the DL models cannot handle such large quality image, thus, they are preprocessed. Specifically, patch sizes of `800x800` are extracted from the original images, now equalling 28029 and 9512 images for training and validation respectively. Figure 2 depicts some samples from the dataset with the bounding boxes drawn on the corresponding category images. As can be seen, the dataset contains images with only a single category, as in Figure 2 lower left (only small vehicles), and several categories within the same image, as in upper left (harbour, small vehicle, swimming pool and ship) or the other two on the right. It is also

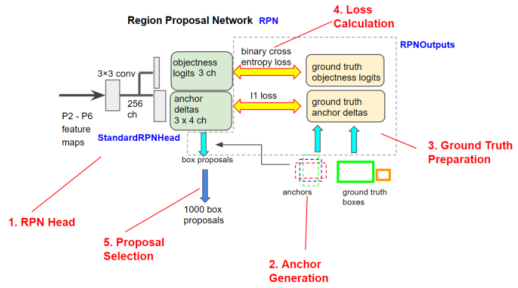


Figure 6: The figure shows the design of the RPN block in the baseline (Adapted from [13]). At each position of the feature map, objectness score of all the anchors (3 by default) at that position as well as there predicted shift is predicted.

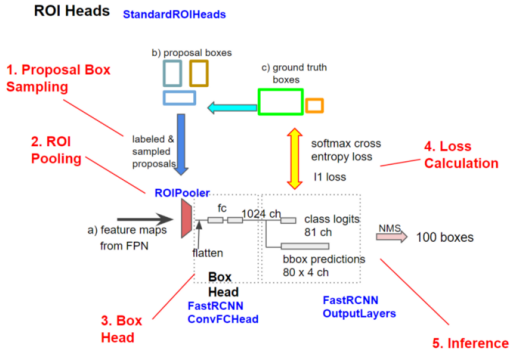


Figure 7: The figure shows the design of the ROI block in the baseline (Adapted from [13]). Each proposed region from the RPN is cropped from the feature map, then an ROIPooler, in our case ROIAlignV2, is used to make all cropped regions have the same spatial size. Then each cropped region is passed through a fully connected layer into two branches, one to output the classes prediction and the other to predict the bounding box shift.

noteworthy that there are multiple instances of the same category within the same image, which is common in the dataset.

The authors list down a few differentiating characteristics of the dataset to contrast it to other ones, claiming that the dataset has:

- a significant quantity of high spatial resolution images
- fifteen categories that are common and important in nature
- a significantly increased number of object instances in each image
- a considerable number of labeled instances per image
- large scale variations of objects (small, medium and large), existing in the same picture in many cases
- data imbalance with respect to objects in images reflecting real-life aerial imaging

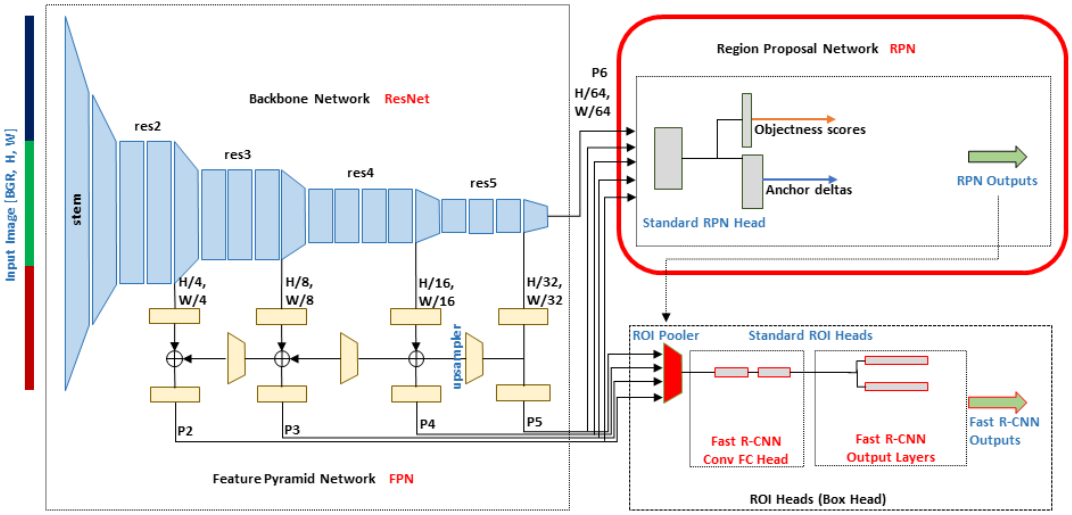


Figure 8: The figure shows the baseline architecture of Faster R-CNN with FPN-based ResNet backbone. Features extracted at different layers of the backbone are passed to the RPN module as well as ROI head module. RPN produces proposals used in the RIO head module to yield final results of the object detection task.

- small sized objects with appearances in ambiguity that requires context to get resolved
- high precision in instance level annotations by experts that are cross-validated by other professionals who followed guidelines for annotations.

To understand the dataset in further details, bounding box information was extracted. The object dimensions are first normalized for this analysis. We plot in Figure 3 height and width of each object bounding box in a correlated fashion. Each color represents a different category. As can be seen in the side bars, most heights and widths (about 80 percent) lie within the range of 0.2 and 0.2 respectively. It indicates that most objects in the images are relatively small, and there are a few that occurs at the highest dimensions.

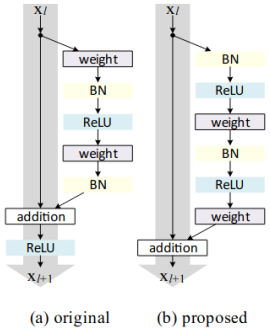
4. Methods

4.1. Baseline

Our baseline for this project is a variation of two-stage object detection model - Faster R-CNN. In the provided dataset, we have objects which vary significantly in scale. To accurately capture all the objects using vanilla Faster R-CNN, we need to have a very high variation in the anchor scales chosen to generate region proposals on the single feature map scale. This task of identifying objects at different scales, and specifically small objects, is very challenging. Furthermore, using images at different scales to overcome this issue is constrained by the memory as well as time consumption while training.

Taking the above statements into consideration, we chose Feature Pyramid Network (FPN) based Faster R-CNN [16] as our baseline model. This network (as shown in Figure 8) exploits the hierarchical property of convolution neural networks (in our case, a ResNet backbone [11]) to extract features at multiple levels rather than taking images at multiple scales. Multi-scale feature maps generated in this manner provides the model with better quality information than the single-level feature map used in the vanilla Faster R-CNN. Figure 4 (d) shows the basic design of the FPN backbone used in our baseline. It can be seen in Figure 4 (d) that predictions are made independently on each level, while also reusing deep features learned by the subsequent layers. The smaller/deeper resolution feature maps are up-sampled (via nearest neighbors) before adding them (element-wise) to larger resolution feature maps as shown in Figure 5.

After getting features at different levels, similar to vanilla Faster R-CNN, Region Proposal Network (RPN) [25] is used as a class-agnostic object detector. The RPN network is originally used on top of a single-scale feature map, but in our case, it will be used on all the multi-scale feature maps generated by our FPN backbone ($P2 - P6$). For each feature map, a 3×3 convolution layer, followed by a separate 1×1 convolution for the objectness score and bounding box regression is applied. As shown in Figure 6, all the feature maps are fed into the RPN block sequentially; the higher resolution feature maps are utilised to generate proposals for small objects while as lower resolution feature maps capture proposals for larger objects. Each scale is used to predict objectness score and anchor deltas for different anchor sizes and at different strides laid on the original image; e.g. [32, 64, 128, 256, 512] anchor sizes with strides [4, 8, 16, 32, 64] corresponding to the features maps $P2 - P6$. At each scale, three different aspect ratios



(a) original (b) proposed

Figure 9: The figure shows the residual flow of input (and the gradient back) in the proposed residual block (b). Note that the straight line is the skip connection here. Adapted from [12].

of the anchors are used to get good proposals for objects of different shapes. The size of the anchors used in the model depends on the size of the objects in the dataset, and thus needs to be tuned to get the best proposals.

After this stage, top-*k* proposal are selected based on the proposals with the highest objectness score at each scale and non-maximum suppression to remove overlapping proposals. These proposals are then utilised by the ROI block to crop the corresponding regions of interests (ROIs) from the feature maps $P2 - P6$. Different methods such as ROI Pooling, ROIAlign and ROIAlignV2 were used to crop the feature maps based on the generated proposals and then resize them to the same size. The ROIs generated in this fashion are then passed to a sequence of fully connected layers as shown in Figure 7 to classify the object within the ROIs and fine-tune the position of the bounding box.

4.2. Proposed Modifications on the baseline

4.2.1 Changing Backbone

In the baseline model, ResNet-101 [11] was used as the backbone model from which we get the multi-scale features used in the subsequent modules of the network. This backbone is comprised of residual blocks which help much better flow of gradients in the backward pass, leading to better convergence for deep neural networks. In [12], a new variant of residual unit was proposed that achieves faster error reduction and a lower training loss. Figure 9 shows the changes done in the residual unit. The authors were able to demonstrate through ablation experiments the smooth propagation of information while training deep neural networks using the proposed residual unit. It also uses group normalization and weight standardization that is useful when the batch size is small. This backbone is pretrained on ImageNet21k. In this context, we will be use the newer version of ResNet-101, known as ResNetv2-101 and analyze the change in the evaluation metric.

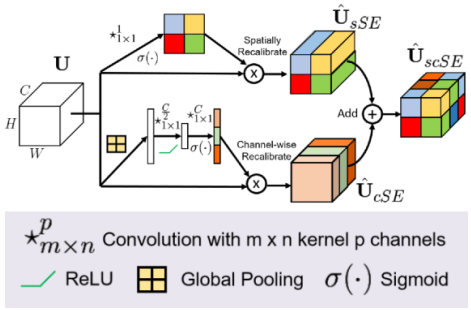


Figure 10: The figure shows the spatial and channel squeeze & excitation block. The upper flow shows the spatial squeeze and excitation, and the lower path shows the channel squeeze and excitation, both of which are then combined together. Adapted from [26].

4.2.2 Deep-RPN block

In order to improve the RPN block to learn more meaningful features for generating better proposals for object and background class, we introduced squeeze and excitation (SE) modules [26] in the RPN block. We use a deeper RPN-head with feature maps recalibrated using concurrent spatial and channel squeeze excitation (scSE) modules to emphasize useful spatial regions and channel present in the feature maps. Using spatial excitation in addition to channel excitation is more useful in our case as having pixel-wise information helps localising objects for better proposals. Figure 10 shows the architectural design for an scSE module.

4.2.3 Data Augmentation

It is common that data augmentations contribute to the improvement in results, fabricating additional data for the network to learn. With that in mind, distinct sets of generic augmentations were assessed for the given task. Resizing shortest edge of objects, horizontal flip, vertical flip, rotation at 90 degrees, brightness adjustment, and saturation were all the different transforms we experimented with. To be accurate, various combinations of these augmentations were applied on the data. Supplementary Table 5 lists all the combinations examined in this work.

4.2.4 Soft-NMS

Due to its suppressing criterion, non-maximum suppression (NMS) is not suitable for detecting objects that are in close proximity to each other. As it takes only the bounding box with the maximum objectness score and suppresses any other box that is above or equal a certain intersection over union (IoU) value N_t with the object.

To address this issue, we opted to use Soft-NMS [2] rather than vanilla NMS in our project. As shown in Figure 11, unlike vanilla NMS, Soft-NMS does not fully sup-

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline+NMS	36.04	56.629	39.293	20.88	43.293	48.695
Baseline+S-NMS	36.397	56.608	39.986	21.139	**43.760**	49.17
Baseline+NMS+A1	35.788	56.875	38.807	21.044	42.645	48.107
Baseline+S-NMS+A1	36.53	57.039	40.11	21.309	43.386	49.331
Baseline+NMS+A2	35.841	56.931	38.888	20.99	42.757	47.601
Baseline+S-NMS+A2	**36.644**	**57.272**	**40.238**	**21.395**	43.726	**49.586**

Table 1: The table shows results of using the baseline with NMS vs Soft-NMS at various anchor sizes. Here, S-NMS stands for Soft-NMS; $A1$ and $A2$ correspond to [16, 64, 128, 256, 512] and [8, 64, 128, 256, 512] anchor sizes respectively.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline-ResNet101	39.65	60.557	43.079	24.11	46.679	54.701
Baseline-ResNetv2-101	**41.977**	**64.562**	**45.931**	**27.555**	**49.479**	**55.971**

Table 2: The table shows the comparison of results of using different backbones. This is to show the effect of the ResNetv2 backbone stand-alone, without any further modifications applied. Note that both backbones use 101 layers in ResNet.

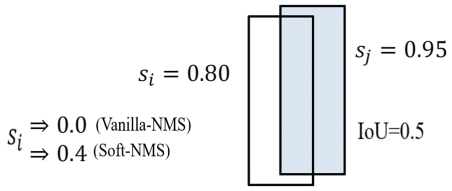


Figure 11: The figure shows the difference between soft-NMS and vanilla NMS with $N_t = 0.5$. In case of vanilla-NMS the second bounding box(s_j) is neglected, while in soft-NMS the bounding box is kept with lower score.

press other bounding boxes; instead, it decays the score as a continuous function in the IoU with the object as follows,

$$s_j = \begin{cases} s_j, & IoU > N_t. \\ s_j(1 - IoU), & IoU \leq N_t. \end{cases} \quad (1)$$

In doing so, we avoid neglecting bounding boxes that actually contains an object but in close proximity to another object. This case occurs for multiple samples in our dataset.

4.2.5 Changing Anchor sizes

The default anchor sizes of $[[32], [64], [128], [256], [512]]$ chosen at each scale in the FPN based Faster R-CNN work well for relatively larger objects as compared to the objects in our dataset. In order for the model to work well on our dataset, we customize the size of the anchors based on the size distribution of the ground truth bounding boxes in our dataset. From Figure 3, we can observe that most of the objects occupy around $15 - 20\%$ of the image area. Taking into account that most of the bounding boxes are small, we decrease the size of the anchors to cover the entire range of small objects in our dataset.

5. Experiments & Results

Evaluation Metrics. Standard COCO metrics [17]: AP (averaged over IoU threshold), AP_{50}, AP_{75}, AP_S, AP_M and AP_L, where S,M and L represent small (area: 10-144 pixels), medium (area:144 to 1024 pixels) and large objects(area: 1024 and above), are used for evaluation.

Experimental Setup We carried out all the experiments with a batch size of 2, and reduced the number of ROI proposals to be passed to ROI Head by a factor of 2 (default is 512) due to the computational constraints of our setup. All of our experiments were trained on a 24GB NVIDIA Quadro RTX 6000 GPU. For finding the best data augmentation, the relevant experiments were trained for $60,000$ and a base learning rate of 0.00025. Later on, all the models were trained for $100,000$ iteration with a base learning rate of 0.0025, which is decreased by a factor of 10 at $50,000$ and $85,000$ iterations. In the experiments, Soft-NMS is utilised, a linearly decaying scoring function is used for reducing the score of detection boxes.

Results of Soft-NMS and Anchor Sizes. Since the most objects in our dataset are small in size, using small anchor sizes was hypothesized to increase the performance of model on small objects. This proved to be true, as shown in Table 1. Here, $A1$ represents [16, 64, 128, 256, 512] and $A2$ represents [8, 64, 128, 256, 512] anchor sizes. From the Table 1, we can observe that the model with smaller anchor sizes, $A1$ or $A2$, lead to a higher AP_S score. However, we can also see that utilizing smaller anchor causes AP_L score of larger objects to fall. To deal with this issue, Soft-NMS comes into play. Table 1 also compares the performance of the baseline model with NMS and Soft-NMS under different anchor sizes. As can be seen, the mean average precision increased from 36.04 to 36.397 when we

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline-ResNetv2 + A2	43.085	66.34	47.585	28.787	49.962	55.218
+ # of DB at 1000	43.715	67.675	48.118	29.33	50.572	55.853
+ NMS thresh at 0.6	43.717	67.432	48.261	29.346	50.558	55.657
Baseline-ResNetv2+Deep-RPN +A2	43.234	66.330	47.486	28.925	50.211	56.937
+ # of DB at 1000	43.928	**67.766**	48.098	29.551	50.942	**57.354**
+ NMS thresh at 0.6	**43.962**	67.544	**48.309**	**29.584**	**50.996**	57.260

Table 3: The table shows the results of using Deep-RPN with spatial and channel squeeze and excitation blocks. Here, $A2$ correspond to [8, 64, 128, 256, 512] anchor sizes; The second and third rows of both table sections mean that the previous model was supplemented with setting the detection boxes at 1000 and then setting the NMS threshold at 0.6.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
BlendV1	43.636	66.965	48.267	29.821	50.957	55.460
+ # of DB at 1000	44.390	**68.485**	48.959	30.497	51.626	56.021
+ NMS thresh at 0.6	44.457	68.245	49.182	**30.574**	51.676	56.077
BlendV2	44.135	66.038	49.110	29.010	52.102	56.983
+ # of DB at 1000	**44.612**	66.798	**49.652**	29.310	**52.751**	**57.443**
+ NMS thresh at 0.6	44.503	66.539	49.581	29.264	52.635	57.228

Table 4: The table shows the final blending of all the modifications on top of the baseline. Here, the second and third rows of all table sections mean that the model mentioned on the first row was supplemented with setting the detection boxes at 1000 and then setting the NMS threshold at 0.6.

use Soft-NMS with default anchor sizes. Using the above anchor sizes ($A1$ and $A2$) improved the mAP even more, with $A2$ and Soft-NMS powered baseline yielding the highest mAP of 36.644. This combination provided the best AP scores across almost all the categories. The experiments reported in Table 1 were trained for $60,000$ iterations with a base learning rate of 0.00025.

Results of Backbone Change. Changing solely the backbone from ResNet-101 to ResNetV2-101, which is supported with a smoother flow of gradients in the skip connections, group normalization, weight standardization and pretrained quality with ImageNet21k, showed a promising boost. Table 2 reveals that the mAP rose to 41.977 from the baseline, with all the other metrics tailing the same trend. The experiments reported in Table 2 were trained for $60,000$ iterations with a base learning rate of 0.0025.

Results of Deep-RPN. Integrating the model RPN with spatial and channel squeeze and excitation blocks, as well as deepening it aided the model to learn more beneficial features, resulting in higher mAP. As can be seen in Table 3, our baseline, empowered with ResNetv2, deep-RPN module and the smaller anchor sizes reached the mAP of 43.962 when the number of detection boxes was set at 1000 and NMS threshold was set to 0.6. A similar increasing pattern was observed with other metrics of mAP. The experiments reported in Table 3 were trained for $100,000$ iterations with a base learning rate of 0.0025.

Results of Blends. Finally, putting all the tweaks onto one place, two versions of what we call blend model are proposed. In Table 4, BlendV1 is our baseline Faster R-CNN with FPN, strengthened with ResNetv2-101 backbone, deeper RPN, smaller anchor sizes ($A2$), and a set of data augmentations. Here, data augmentation set comprises resize shortest edge, horizontal flip, vertical flip and saturation, which provided the highest mAP values as reported in Supplementary Table 5. BlendV2 is the same model with an addition of Soft-NMS. BlendV1 computed at 1000 detection boxes with NMS threshold at 0.6 reached 44.457 mAP, and BlendV2 threholded at 1000 detection boxes landed at the highest of 44.612 mAP. Note that the small objects precision was high with BlendV1, however, medium and large object suffered a small decrease. This issue was resolved with BlendV2 with the help of Soft-NMS.

Qualitative Results. The Figures 12, 13, and 14 show the original image on the left and predicted detection boxes drawn on them on the right. As can be seen, the model outputs detection boxes even when the objects are small very accurately.

6. Discussion and Conclusion

This paper presents a set of techniques that can be applied to boost the network, Faster R-CNN in particular, when trained for aerial images. With an extensive analysis, we show that the baseline model can dress up with certain additional features such as the backbone change, data aug-

Figure 12: The figure shows output from BlendV2.

Figure 13: The figure shows output from BlendV2.

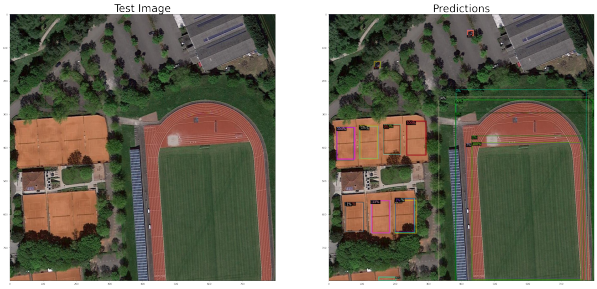


Figure 14: The figure shows output from BlendV2.

mentations, SE-based deeper RPN and soft NMS to reach higher results. Such a huge jump in the mAP proves that the integration of such components to the network can benefit it to a great extent. Utilizing spatial and channel squeeze and excitation block in the RPN as well as increasing the layers of the RPN are considered as our contributions of the work, as a cherry on top of all the other feature combinations. The loss function, an integral part of the neural networks, can be of tremendous help too, if properly chosen and tuned. In our case, the loss functions we experimented with (as described in details in Supplementary section) did not contribute to the success of the network. We believe it is due to the fact that the loss functions were not fine-tuned for the current task, and with more experiments, a reasonable improvement may be achieved. We hope to further investigate a better classification and regression loss suited for countering the challenges of significant scale variation and the class imbalance problem prevalent in the task.

References

[1] A. Bochkovskiy, C. Wang, and H. M. Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020. 2

[2] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis. Soft-nms – improving object detection with one line of code, 2017. 5

[3] J. Cao, H. Cholakkal, R. M. Anwer, F. S. Khan, Y. Pang, and L. Shao. D2det: Towards high quality object detection and instance segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11482–11491, 2020. 2

[4] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding, 2016. 2

[5] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1, 2005. 1

[6] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 2

[7] D. Du, Y. Qi, H. Yu, Y. Yang, K. Duan, G. Li, W. Zhang, Q. Huang, and Q. Tian. The unmanned aerial vehicle benchmark: Object detection and tracking, 2018. 2

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 2

[9] R. B. Girshick. Fast R-CNN. *CoRR*, abs/1504.08083, 2015. 2

[10] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013. 2

[11] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015. 4, 5

[12] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *CoRR*, abs/1603.05027, 2016. 5

[13] H. Honda. Digging into detectron2 -part4. https://bit.ly/31xHBnQ, October. Accessed: 2021-10-7. 3

[14] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord. xview: Objects in context in overhead imagery, 2018. 3

[15] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han. Object

detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:296307, Jan 2020. 3

[16] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 3, 4

[17] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollr. Microsoft coco: Common objects in context, 2015. 2, 6

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 2

[19] J. Nie, R. M. Anwer, H. Cholakkal, F. S. Khan, Y. Pang, and L. Shao. Enriched feature guided refinement network for object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9537–9546, 2019. 2

[20] Y. Pang, T. Wang, R. M. Anwer, F. S. Khan, and L. Shao. Efficient featurized image pyramid network for single shot detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7336–7344, 2019. 2

[21] K. Ranasinghe, M. Naseer, M. Hayat, S. H. Khan, and F. S. Khan. Orthogonal projection loss. *CoRR*, abs/2103.14021, 2021. 10

[22] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 2

[23] J. Redmon and A. Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016. 2

[24] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. 2

[25] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. 1, 2, 4

[26] A. G. Roy, N. Navab, and C. Wachinger. Concurrent spatial and channel squeeze & excitation in fully convolutional networks. *CoRR*, abs/1803.02579, 2018. 5

[27] L. W. Sommer, T. Schuchert, and J. Beyerer. Fast deep vehicle detection in aerial images. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 311–319, 2017. 2

[28] T. Wang, Y. Zhu, C. Zhao, W. Zeng, J. Wang, and M. Tang. Adaptive class suppression loss for long-tail object detection, 2021. 11

[29] S. Waqas Zamir, A. Arora, A. Gupta, S. Khan, G. Sun, F. Shahbaz Khan, F. Zhu, L. Shao, G.-S. Xia, and X. Bai. isaid: A large-scale dataset for instance segmentation in aerial images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 28–37, 2019. 2, 3

[30] G.-S. Xia, X. Bai, J. Ding, Z. Zhu, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang. Dota: A large-scale dataset for object detection in aerial images, 2019. 3

[31] W. Xiang, D. Zhang, V. Athitsos, and H. Yu. Context-aware single-shot detector. *CoRR*, abs/1707.08682, 2017. 2

[32] F. Yang, H. Fan, P. Chu, E. Blasch, and H. Ling. Clustered object detection in aerial images. *CoRR*, abs/1904.08008, 2019. 2

[33] S. S. A. Zaidi, M. S. Ansari, A. Aslam, N. Kanwal, M. N. Asghar, and B. A. Lee. A survey of modern deep learning based object detection models. *CoRR*, abs/2104.11892, 2021. 1

[34] F. Zhang, X. Wang, S. Zhou, and Y. Wang. Dardet: A dense anchor-free rotated object detector in aerial images. *IEEE Geoscience and Remote Sensing Letters*, page 11, 2021. 2

[35] X. Zhu, S. Lyu, X. Wang, and Q. Zhao. Tph-yolov5: Improved yolov5 based on transformer prediction head for object detection on drone-captured scenarios. *CoRR*, abs/2108.11539, 2021. 2

[36] I. evo and A. Avramovi. Convolutional neural network based automatic object detection on aerial images. *IEEE Geoscience and Remote Sensing Letters*, 13(5):740–744, 2016. 2

Augmentations	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Resize SE, HFlip (baseline)	35.284	56.698	37.901	20.609	42.403	46.621
Resize SE, HFlip, VFlip	35.457	56.761	38.488	20.481	42.433	47.864
Resize SE, HFlip, Rotation(90)	33.679	53.569	36.644	19.733	40.206	45.522
Resize SE, HFlip, VFlip, Brightness	35.356	56.596	38.641	20.797	42.146	47.303
Resize SE, HFlip, VFlip, Saturation	**36.181**	**56.967**	**38.736**	**21.420**	**42.979**	**48.428**

Table 5: The table shows the results obtained from using various data augmentations.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline-ResNetv2	43.06	65.715	47.52	28.3	51.143	56.81
+ # of DB at 1000	43.414	66.409	47.885	28.512	51.549	**57.275**
+ NMS thresh at 0.6	43.466	66.282	48.021	28.561	**51.618**	57.22
Baseline-ResNetv2+A2	43.085	66.34	47.585	28.787	49.962	55.218
+ # of DB at 1000	43.715	**67.675**	48.118	29.33	50.572	55.853
+ NMS thresh at 0.6	**43.717**	67.432	**48.261**	**29.346**	50.558	55.657

Table 6: The table shows the results of thresholding detection boxes at 1000 and NMS at 0.6.

7. Supplementary Materials

7.1. Further Analysis on Dataset

To further investigate the dataset, the areas of object instances are calculated and the square root of them are derived to understand the average frequency of such areas in different categories. Figure 15 shows the frequency of instances having similar areas of objects, indicating that the majority of objects have a square root of an area under 100. This is used to give a sense of measure to the dimensions of objects and it is useful to provide information on what sizes of proposals can be utilized when training. Similarly, the calculations are done for each category to see the range of different size variations, as depicted in Figure 16. Note that the axes have different ranges here as this was the only noticeable way to plot them in a comparative manner. The same pattern is shown here that each category holds the highest count of square root of areas that are under around 100.

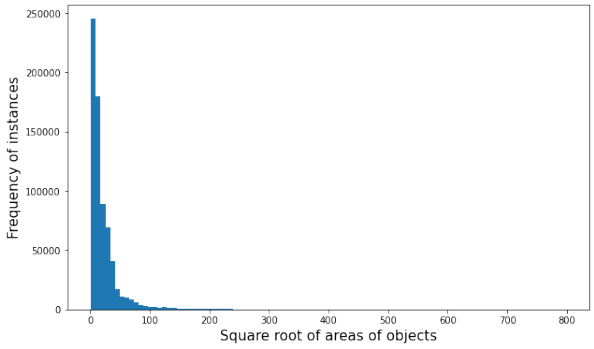


Figure 15: The figure shows frequencies of instances (y-axis) having the square root of areas of objects (x-axis).

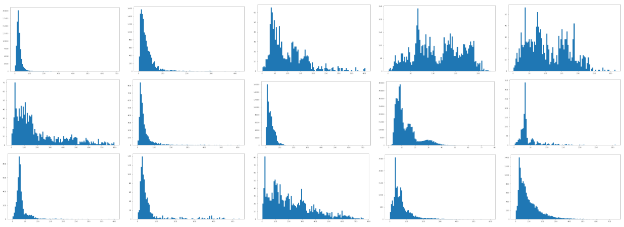


Figure 16: The figure shows frequencies of instances (y-axis) having the square root of areas of objects (x-axis) for each category.

7.2. Detailed Experiments & Results

7.2.1 Loss Functions

Orthogonal Projection Loss. Recently, a new loss function was proposed for the task of classification, known as Orthogonal Projection Loss (OPL)[21]. This loss function complements the cross entropy loss in having well separated features for different classes. OPL imposes orthogonality constraints on the mini-batch level, forcing inter-class separation as well as intra-class clustering. We use this objective function for the purposing of aiming a better discrimination between the background and foreground classes which can further enhance the prediction of our model.

Consider F be a deep neural network, which can be separated into F_ψ and F_ϕ which are the feature extraction block and the classification head respectively. Given an input-output pair $\{x_i, y_i\}$, $f_i = F_\psi(x_i)$ will be the intermediate features of the network. The OPL enforces the features belonging to different classes to be orthogonal to each other, while features from the same class should be similar. The resulting loss function on a batch of images B is :

$$s = \sum_{\substack{i,j \in B \\ y_i = y_j}} < f_i, f_j > \tag{2}$$

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline-ResNetv2-50	42.617	64.627	47.255	28.033	50.448	56.311
+ # of DB at 1000	43.063	65.294	47.748	28.356	51.002	56.954
+ NMS thresh at 0.6	42.970	65.023	47.708	28.306	50.897	56.724

Table 7: The table shows the results obtained by changing the backbone from ResNetv2-101 to ResNetv2-50.

Method	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Baseline-ResNetv2	43.06	65.715	47.52	28.3	51.143	56.81
+ # of DB at 1000	43.414	66.409	47.885	28.512	51.549	57.275
+ NMS thresh at 0.6	**43.466**	**66.282**	**48.021**	**28.561**	**51.618**	**57.22**
Baseline-ResNetv2+OPL	43.038	65.521	47.308	27.954	50.861	56
+ # of DB at 1000	43.371	66.177	47.635	28.162	51.284	56.464
+ NMS thresh at 0.6	43.389	66.01	47.702	28.184	51.363	56.328
Baseline-ResNet	36.099	57.259	38.934	21.256	43.518	49.393
ACLS (0.7 threshold)	29.162	46.907	31.041	17.645	35.193	39.933
ACLS (0.5 threshold)	30.588	49.475	32.689	18.105	37.622	41.683
ACLS (0.4 threshold)	31.946	51.217	34.095	18.725	38.929	43.583
ACLS (0.3 threshold)	32.123	51.628	34.273	18.508	39.081	44.197

Table 8: The table shows the results of using Orthogonal Projection Loss (OPL) and Adaptive Class Suppression Loss (ACSL). OPL results were evaluated on ResNetv2 backbone, while ResNet backbone was used in ACLS experiments.

$$d = \sum_{\substack{i,k \in B \\ y_i \neq y_k}} < f_i, f_k > \tag{3}$$

$$L_{OPL} = (1 - s) + |d| \tag{4}$$

where $< ., . >$ is the cosine similarity operator and $|.|$ is the absolute value operator. So, our overall loss classification loss is the combination of CE and OPL loss:

$$L = L_{CE} + \lambda * L_{OPL} \tag{5}$$

where λ is a hyper-parameter to control the effect of the OPL loss.

OPL Results. Table 8 shows the results of baseline with ResNetv2 with and without OPL loss. As is evident, the introduced loss function did not perform well in any part of the metric. This can be on account of the small object sizes and overlapping feature our dataset holds. Although the loss function is capable of discriminating between fore- and background objects from each other, small area coverage by objects and them overlapping each other in many cases may be causing the objective function confuse, therefore, yield unsatisfactory results.

Adaptive Class Suppression Loss. In order to deal with the class imbalance in our dataset, which is known as long tail problem, we replaced the vanilla cross-entropy loss with Adaptive Class Suppression Loss (ACSL)[28]. Cross-entropy goal is to output a hot one confidence vector with one at the correct class prediction and zero otherwise. To accomplish this, it trains the classifier to output negative suppression gradients for negative class to output low confidence for this classes, this would be useful in case of a balanced dataset, but in case of long tail datase, some classes rarely appear so the model will learn to always suppress them. ACSL addresses this problem , as shown in Eq.6 by adding a parameter w to only output negative suppressing gradients for the negative classes that causes confusion, i.e. have high confidence, and does not suppress other negative classes.

$$L_{ACSL}(x_s) = -\sum_{i=1}^{C} w_i log(p_i) \tag{6}$$

In case of the positive class, w_i is 1 so the positive gradient flows normally. In case of the negative class w_i is set to zero if the confidence is lower than a threshold η to negate the suppression gradient, otherwise it is set to 1. This is expressed in the following equation,

$$w_i = \begin{cases} 1, & \text{if } i = k \\ 1, & \text{if } i \neq k \text{ and } p_i \geq \eta \\ 0, & \text{if } i \neq k \text{ and } p_i < \eta \,. \end{cases} \tag{7}$$

Also classes are split into three groups, according to the number of samples, which are:rare,common and frequent, and percentage of background samples are suppressed in case of rare and common classes. Only 1% of background samples are taken into account in ca se of rare class and 10% in case of common class.

ACSL Results. Table 8 shows the results of baseline with ResNet101 with and without ACSL loss. As shown, the introduced loss function did not perform well in any part of the metric. This can be on account for inaccurate hyperparameters or split of classes which required further analysis that the time did not allow.

7.2.2 New Backbone

To make further comparisons on the backbone of the Faster R-CNN network, instead of using ResNetv2-101, we experimented with ResNetv2-50 as a backbone to see how much it would affect in terms of accuracy. To perform this experiment, we are using our BlendV2, the best performing model, only changing the backbone to ResNetv2-50. Table 7 shows that the results suffered a small decrease with the introduction of the ResNetv2-50, a smaller network. It proves that the model backbone we chose for BlendV1 and V2 are crucial for the whole network to reach such a score.