Mafat (DDR&D) Challenge
Detection and fine grained classification of objects in aerial imagery

Aviad Moreshet
Bar Ilan University

June 21, 2018

Abstract

Object detection is an important and challenging problem in computer vision field. Despite the major advances in detection of objects in natural scenes, not much work has been done on aerial imagery scenes. Those scenes are characterized by a unique bird's eye view, tilted orientation, may contain up to dozens or hundreds of objects with a relatively small objects, etc. In this paper I tried to solve Mafat Challenge of Detection and fine-grained classification of objects in aerial imagery, by training and fine-tuning several ConvNets architectures on the published dataset.

1 The Task

There are mainly four types of tasks in the field of Object Recognition in Computer Vision: Object Classification and Localization, in which we need to classify whether an object appears in an image or not, along with a surrounding bounding box. Object Detection, which is like the previous task but now multiple objects are allowed per image. Semantic Segmantation, in which we need to label (usually by coloring) each pixel in an image with a category label, and finally Instance Segmentation, in which we need detect multiple objects per image and also color (label) their exact pixels.
In this challenge, the task is in the Object Detection field. More precisely, we need to detect 4 coarse grained classes and perform fine grained classification of objects in aerial imagery.
The coarse grained classes are Small Vehicle, Large Vehicle, Solar Panel and Utility Pole. Small and Large Vehicle classes have further fine grained features we need to detect, as following:

1. Small Vehicle
   (a) Subclasses: Sedan, Hatchback, Minivan, Van, Pickup truck, Jeep, Public vehicle.
   (b) Features: Sunroof, Taxi, Luggage carrier, Open cargo area, enclosed cab, Wrecked, Spare wheel.
   (c) Colors: Yellow, Red, Blue, Black, Silver/Grey, White, Other.
2. Large Vehicle
   (a) Subclasses: Truck, Light Truck, Concrete mixer truck, Dedicated Agricultural vehicle, Crane truck, Prime mover, Tanker, Bus, Minibus.
   (b) Features: Open cargo area, Vents, Air conditioner, Wrecked, Enclosed box, Enclosed cab, Ladder, Flatbed, Soft shell box, Harnessed to a cart.
   (c) Colors: Yellow, Red, Blue, Black, Silver/Grey, White, Other.

In the detection of Subclasses and Colors classes, each detection should include a single value, while the Features detection has to include multiple values (has sunroof? has luggage carrier? etc).

2 Evaluation

The final score will be a multiplication of the Coarse Grained classes detection score and the Fine Grained classes classification score.

2.1 Coarse Grained Score

Here we use the popular Mean Average Precision score. First, we have to define TP and FP of a bounding box prediction. TP will be defined by the IoU (intersection area over union area) score. If the IoU between observed bounding box and predicted bounding box is larger than 0.25, we mark this detection as TP. For Utility Poles which are labeled by 1 point, if the distance between observed and predicted point is less than 30 pixels, we consider the detection as TP as well. Otherwise, a detection is marked as FP. For multiple detections, only the first detection is considered TP. The rest will be considered FP. Now, we can calculate AP:

$$AP(class) = \frac{1}{K} \sum_{k=1}^{K} precision(k) rel(k)$$

$$mAP = \frac{1}{N_c} \sum_{class=1}^{N_c} AP(class)$$

Where K is the total number of objects from the class in the test data, $precision(k)$ is the precision calculated over the first k objects, $rel(k)$ is 1 if object k is true, else 0 and N_c is the number of categories.

2.2 Fine Grained Score

Calculated only for Coarse Grained classes that were predicted correctly. The challenge suggested to calculate the score using Cohen's Kappa score, however, due to lack of time to classify all the fine grained features, I decided to evaluate my results (of the several classes I did classify) with a normal precision evaluation.

3 The Dataset

3.1 Brief

The dataset contains thousands of aerial images, taken from diverse geographical locations, different times, resolutions, area coverage and photo conditions (weather, angles and lighting). The resolutions vary between 5cm to 15cm Ground Sample Distance (the distance between pixel centers measured on the ground). Some samples (generated with VoTT [1]):

3.2 Analysis

Most of the dataset does not contain objects at all. Only 51 images are fully tagged. 2161 images are partially tagged, 7121 images were verified to contain no objects at all, and 36 images are untagged.
Regarding labels, the dataset definitely biases towards small vehicles, which appear more than all of the other classes combined. Distribution can be seen in Figure 1. As mentioned before, large and small vehicles are further tagged with

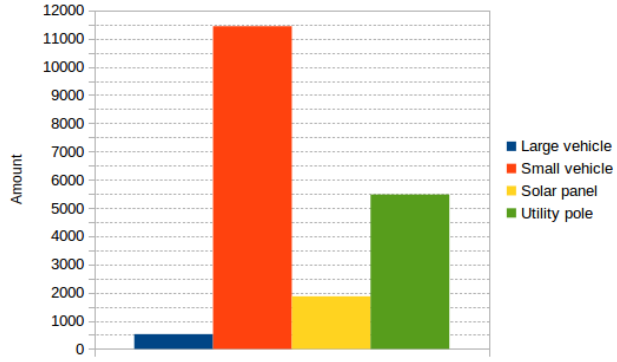

Figure 1: Dataset Coarse Grained Labels Distribution

fine-grained labels, consisted of 3 categories: subclass, feature and color.
The large vehicle class has only hundreds of subclass tags, while small vehicle has many thousands, as seen in Figure 2. The same goes with feature tags, as

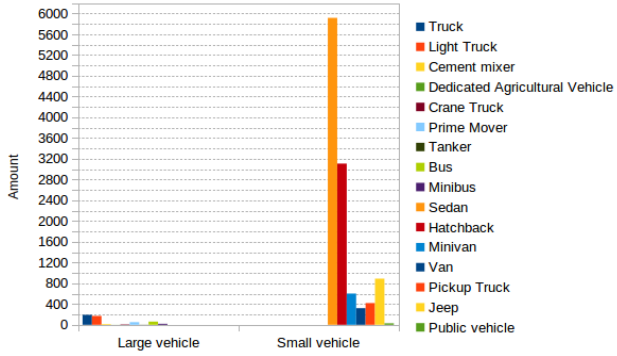

Figure 2: Dataset Subclass Labels Distribution

seen in Figure 3, and also with color tags, as seen in Figure 4.
On average, there are 8 objects in an image, while the most crowded image contains 164 objects altogether.

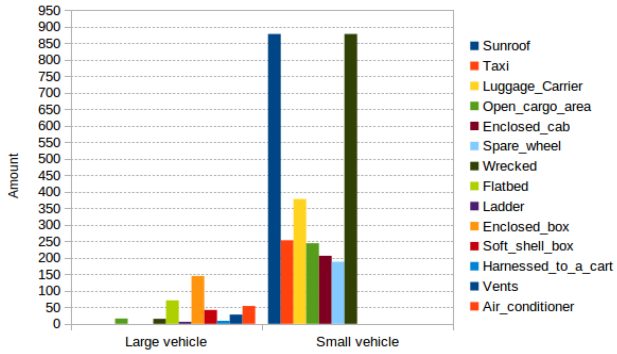

Figure 3: Dataset Feature Labels Distribution

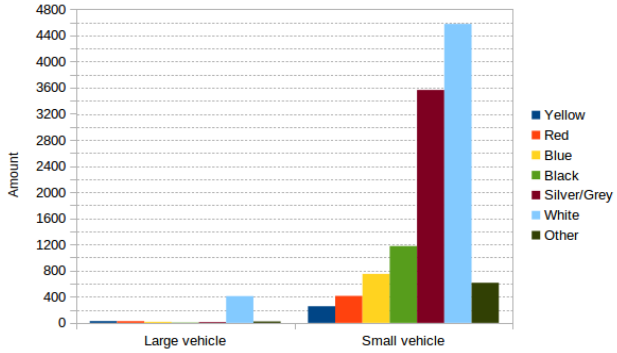

Figure 4: Dataset Color Labels Distribution

In Figure 5 we can see just how small are our objects, and what is their frequency in terms of square rooted area of pixels. The large vehicle objects spans roughly on all the size range, with more appearances around sizes 40-100. Small vehicle objects, however, mostly appear in sizes 20-70. Lastly, our solar panel objects are relatively very small, ranging around sizes 10-40.

3.3 Dataset splits

I chose to randomly select 90% of the dataset as training set and the rest 10% as validation set. Test set should be supplied by the competition organizers.

3.4 Dataset formats

I wrote a converter of the dataset into ImageNet [4] and Pascal VOC [5] formats in order to feed it to the neural networks which support only popular well-known formats.

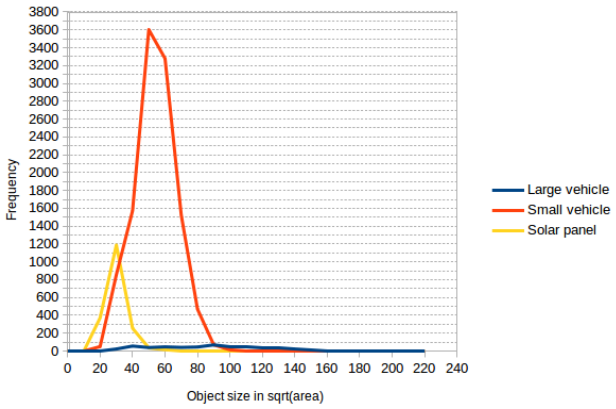

Figure 5: Dataset Objects Size Distribution

4 Chosen Neural Networks

Following the DOTA paper [2] in which Faster-RCNN [6] achieved best scores on the detection task on an aerial dataset, I decided to use it on my aerial dataset as well. The F-RCNN [6] backbone network was an ImageNet [4] pretrained ResNet-101 [3]. I used Jianwei Yang's implementation [8] of this architecture in PyTorch [7].
For classification of fine-grained features I used an ImageNet [4] pretrained ResNet-50 [3], which should give a good trade-off between speed and accuracy (ResNet-101 and ResNet-152 would give a little more accuracy but take longer time to train). I used the default PyTorch [7] implementation for this architecture.

5 Challenges

1. Oriented bounding box
The dataset consisted of oriented bounding boxes (bounding boxes which are not axis aligned).
The popular dataset formats however support axis aligned bounding boxes only, called horizontal bounding boxes. In addition, OBBs are more difficult to learn than HBBs since they are more strict. Hence I decided to wrap each OBB in an HBB, essentially making all bounding boxes axis aligned.

2. Classes annotated with a single point
The utility pole class objects are annotated with a single point instead of 4 points bounding box. Therefore, in the initial experiments I've decided to remove all the utility poles objects from the dataset (and also images which contained only utility poles, 533 in total), as they are extremely difficult to detect and posed

several problems in the training. Later, I added them to the training data by extending each point to a 1x1 bounding box.

3. Fine Grained features
Fine grained features classification is not part of the standard detection ConvNets implementations, which usually classify only one set of classes. In order to allow the F-RCNN [6] to learn fine grained features, I decided to integrate several Fully Connected layers on top of the network which would learn each of the fine grained features, starting from the color feature. Trying to integrate the additional layers in the original network implementation was difficult and accompanied by weeks of debugging the network and trying to figure out why it doesn't learn the fine grained features, or worse, why does it badly ruin the scores of other coarse grained classes.
I had to solve many size mismatch bugs, re-compile some CUDA low level modules which contained hard-coded tensor sizes, etc. I have documented the results of those trials but did not add them to my experiments list as they have yielded really low scores.
I realized that perhaps the network doesn't learn the fine grained features along with the coarse grained ones because in order to learn a good representation for the coarse grained classes, it omits many details (such as color) for this task. To prove this, I trained a separate ResNet-50 [3] for the color fine grained feature, which gave good results supporting my assumption. I hereby decided to take this route and train this network for several more fine grained features.

4. Falsely annotated images
Some image annotations contained bounding boxes with coordiantes outside of the image boundaries. I decided to crop these coordinates to fit inside the images, which in my opinion better than entirely removing them from the dataset, and might also even help, as a kind of regularization.

5. Small objects
The objects appeared in the images are much smaller than standard objects networks usually try to detect. Therefore, the Region Proposal Network of the F-RCNN [6] initially missed all the objects. I had to fine-tune some RPN related hyper-parameters in order to be able to detect the small objects in the dataset. The most important parameter was to set anchor scales to [1, 2, 4, 8, 16], which helped covering various objects sizes.

6. High resolution images
About 100 images in the dataset have relatively high resolutions reaching up to 4k. Including those images in the dataset incurred a big downscaling, since they wouldn't fit naturally into GPU memory. This big downscaling is later shown to hurt Large Vehicle AP.

6 Experiments

I mostly used default learning-related hyper parameters in the following experiments, and fine-tuned the hyper parameters which would help the network to better learn the small sized objects of the dataset. There are more than 30 hyper parameters, so I'll mention the most important ones, and of course each experiment will describe the hyper parameters tuned for the experiment.
For the F-RCNN [6], learning rate was 0.001, optimizer was SGD with 0.9 momentum, weight decay was 5e-4, epochs number was 15, RPN anchor ratios were [0.5,1,2] and RPN feature stride was 16. Basic augmentation of horizontal flip was applied. Learning rate scheduler policy was to divide learning rate by 10 every 5 epochs.
As for the ResNet-50 [3], optimizer was also SGD, batch size was 256, learning rate 0.1, momentum was 0.9 and weight decay was 1e-4. The network was trained for 90 epochs. Augmentations used were of horizontal flip and random cropping. Learning rate scheduler policy was to divide learning rate by 10 every 30 epochs.

The initial experiments results (before I found the baseline hyper paremeters setup) are omitted, because the network didn't even train. For the same reason, experiments which contained bugs (due to the error and trial nature of the process) are also omitted.

In addition, I did not see reasons to collect training plots regarding loss and score metrics until I started optimizing them, since the first upcoming experiments have other targets, and the major challenge and time-consuming tasks of the project was to achieve results on this difficult unique dataset.

6.1 Experient 1

This is the first experiment which yielded actual results.
I used only images up to 1k resolution, with 2 classes: Small Vehicle and Large Vehicle (Solar Panel did not appear in filtered dataset). Scale used was 600 (which didn't basically affect anything since all the images were around 1000x600), and batch size was 2.
Results:
Large Vehicle AP = 0.7531
Small Vehicle AP = 0.8962
mAP = 0.82465.

6.2 Experient 2

In this experiment the exact same parameters were used, but now the training consisted of all the images. The scale parameter now took effect and actually rescaled all the large images small side to be 600 (i.e. 4000x2500 image would

now be 960x600).
Results:
Large Vehicle AP = 0.3520
Small Vehicle AP = 0.7978
Solar Panel AP = 0.0529
mAP = 0.4009

6.3 Experiment 3

Again, same parameters were used, but now I wanted to test the effect of the scale size on the learning. Therefore, I used the maximum scale size I could (due to 12GB GPU memory restriction), which was 1600.
Results:
Large Vehicle AP = 0.4987
Small Vehicle AP = 0.8917
Solar Panel AP = 0.6431
mAP = 0.6778

6.4 Experiment 4

I now wanted to test a combination the scale parameter values, thus, I set the scale size to be [600, 1600] (meaning each image is randomly scaled to either of those sizes). All other parameters remained the same.
Results:
Large Vehicle AP = 0.3340
Small Vehicle AP = 0.7906
Solar Panel AP = 0.1700
mAP = 0.4316

6.5 Experiment 5

I've returned to Experiment 3 parameters, and added the Utility Pole class to the training, by generating a 1x1 bounding box for each Utility Pole point.
Results:
Large Vehicle AP = 0.5416
Small Vehicle AP = 0.9049
Solar Panel AP = 0.4545
Utility Pole AP = 0.2363
mAP = 0.5343

6.6 Experiment 6

In the challenge, if the IoU between observed bounding box and predicted bounding box is larger than 0.25, a prediction is considered TP. However in the above experiments I used 0.5 as the IoU threshold since I believe 0.25 is too low. In this experiment I used the exact same parameters as above, with 0.25

IoU. Results:
Large Vehicle AP = 0.6536
Small Vehicle AP = 0.9090
Solar Panel AP = 0.6364
Utility Pole AP = 0.2798
mAP = 0.6197

6.7 Experiment 7

I now trained a ResNet-50 [3] network from scratch for classifying the color fine grained feature of Small and Large Vehicles.
For this task I cropped all tagged Small and Large Vehicles and fed them to the network for classification of the color.
I received a 75.836% precision on this feature.

6.8 Experiment 8

I now wanted to check whether fine-tuning a pretrained ResNet-50 [3] would encourage transfer-learning and help to improve the classification precision. I managed to improve the precision up to 79.348%.

6.9 Experiment 9

Now I trained (from scratch) a ResNet-50 network for classifying the Subclass fine grained feature of Small Vehicles only (which differs from Large Vehicle subclasses). The network however received a disappointing 51.399% precision on this feature.

6.10 Experiment 10

I now repeated the exact previous experiment but with a pretrained ResNet-50. Just like in the previous experiments the pretrained network beat the other fully trained network, but with a small gap. The network reached 52.710% precision on this feature.

6.11 Experiment 11

This time I fine-tuned a pretrained ResNet-50 on the Large Vehicle fine grained Subclass feature. Surprisingly, the model reached a precision of 45.283%, despite having a relatively small ($\approx$500) train examples.

7 Experiments Conclusions

7.1 Coarse grained classes

The F-RCNN [6] network have successfully learned to detect the Small Vehicle class with AP of 0.91. It also succeed to detect Large Vehicle class with 0.75 AP in the first experiment despite having only hundreds of samples of it. The AP decreased to 0.54 (0.65 with the Challenge's IoU) when more classes and high resolution images were added. The decrease in the results can be explained in several ways. First, the network may have confused the Large Vehicle with other classes, such as Solar Panel. Second, it is reasonable that the down scaling process of the high resolution images added even smaller and perhaps skewed objects, which were difficult for the network to correctly detect and classify. This theory is supported by experiment 3, which shows that less downscaling increases AP. This issue actually affects all the results altogether.
The network detected Solar Panel class with 0.63 AP. The Solar Panel class average size is as almost as the size of the Small Cars, but appears fifth times less in the dataset. Utility Pole class was detected with 0.27 AP, despite it's small size, probably because it had around 6000 tagged instances in the dataset.
Another aspect to touch is transfer learning. We can conclude that despite the fact that the network's ResNet [3] backbone is trained on ImageNet [4], which contains images in different angles, perspectives and views, the network manages to achieve significant results on the relatively small dataset here, thus effectively performing transfer learning.

7.2 Fine grained classes

In all the experiments fine-tuning the ResNet-50 [3] yielded better results than training it from scratch, which indicates a useful transfer learning. The network successfully classified the Color feature with 79.34% (Large and Small Vehicles combined), the Subclass feature of Small Vehicles with 52.71% and the Subclass feature of Large Vehicles with 45.28%. The difference between the pretrained and fully trained networks is unsurprising, and can probably be explained due to the relatively small object instances fed to the network. The ImageNet [4] dataset, on which the pretrained network is trained on, consists of 14 million images, in contrast to the several thousands we got here.
The difference between the Color and Subclass precisions can also be explained due to the difficulty of learning characteristics and nuances between different cars subclasses, unlike colors, which usually fill a significant part of the image.

8 Further Work

Regarding the ≈100 high resolution images which required a big downscaling, perhaps with more time I could implement a better solution:
Perform cropping to smaller sizes, thus avoiding a big downscaling. We would need to take care of two problems:

1. Splitted objects. To remediate this problem, for each two crops, we would take a third crop in-between, to cover splitted objects.
2. Duplicate objects. This would be easily remediated using Non Maximum Suppression algorithm, which would remove the duplicated objects.

More augmentations could have been made, either for all the classes and especially for the lower frequency ones.
For the fine-grained classification, perhaps more basic augmentations would suffice: scaling, rotating, translating, lighting, etc.
For the detection task in general that would be a bit more problematic since we have many objects per image and we need to take care about bounding boxes. A creative paper named Cut, Paste and Learn [9] suggests a unique way to generate more images using crops of our objects on realistic scenes. We could as well utilize here all the empty images we have in our dataset, as well as benefit of increased learning over small objects, as shown in the paper.

I also did not have enough time to train the ResNet-50 network on all the fine grained features and calculate a cumulative score for the fine grained task which takes into account all the Cohen Kappas' scores of all the features, but only managed to proof the feasibility of this task by training on the Color and Subclass features.

References

[1] Visual Object Tagging Tool: An electron app for building end to end Object Detection Models from Images and Videos.
`https://github.com/Microsoft/VoTT`

[2] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, Liangpei Zhang. *DOTA: A Large-scale Dataset for Object Detection in Aerial Images*. arXiv:1711.10398v2 [cs.CV], 2018.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. *Deep Residual Learning for Image Recognition*. 10.1109/CVPR.2016.90, 2016.

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, Li Fei-Fei. *ImageNet: A large-scale hierarchical image database*. 10.1109/CVPR.2009.5206848, 2009.

[5] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, Andrew Zisserman. *The Pascal Visual Object Classes (VOC) Challenge*. 10.1007/s11263-009-0275-4, 2010.

[6] S. Ren, K. He, R. B. Girshick, and J. Sun. proposal networks. *Faster R-CNN: towards real-time object detection with region*. 10.1109/TPAMI.2016.2577031, 2017.

[7] PyTorch is a deep learning framework for fast, flexible experimentation.
https://pytorch.org/

[8] Jianwei Yang's F-RCNN PyTorch implementation.
https://github.com/jwyang/faster-rcnn.pytorch

[9] Debidatta Dwibedi, Ishan Misra, Martial Hebert. Cut, Paste and Learn: Surprisingly Easy Synthesis for Instance Detection.
arXiv:1708.01642v1 [cs.CV]