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

Aviad Moreshet
204311112

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 Subclass 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 Bouding 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. We'll calculate the score using Cohen's Kappa. The Kappa score is calculated separately for the subclass, the color and each feature of the main category. Then, the category score is an average of the Kappa of each feature. Finally, the Fine Grained score is a weighted average between the score of the small vehicles and the large vehicles.

$$score_{small} = \frac{1}{M_{small}} \sum_{i}^{M_{small}} k_i$$

$$score_{large} = \frac{1}{M_{large}} \sum_{i}^{M_{large}} k_i$$

Where k_i is Cohen's Kappa for each feature i, M_{small} is the number of features of the small vehicles, and M_{large} is the number of features of the large vehicles (including the subclass and color). Then, the total Fined Grained score is:

$$score = \frac{N_{large}score_{large} + N_{small}score_{small}}{N_{large} + N_{small}}$$

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:

3.2 Analyses

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 the other classes, combined. Distribution can be seen in Figure 1. As mentioned before, large and small vehicles are further tagged with fine-

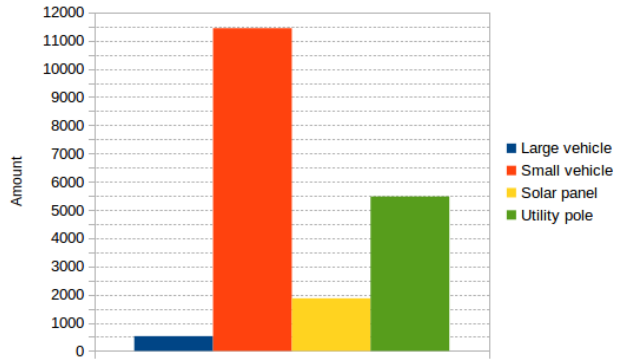

Figure 1: Dataset Coarse Grained Labels Distribution

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

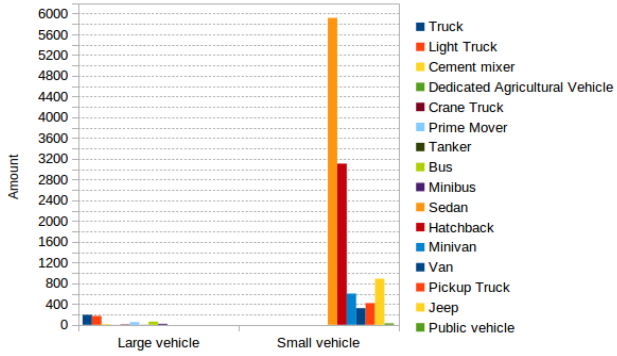

Figure 2: Dataset Subclass Labels Distribution

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.

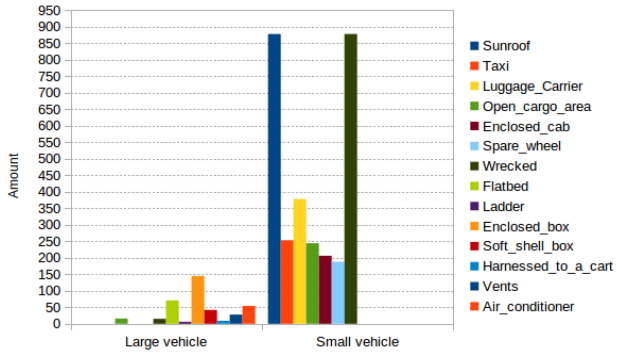

Figure 3: Dataset Feature Labels Distribution

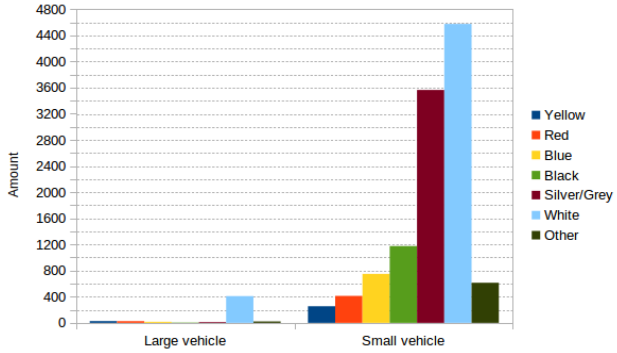

Figure 4: Dataset Color Labels Distribution

¡Complete chart regarding objects frequency per image, make bins and then calculate frequencies¿

In Figure 5 we can see just how small are our objects, and what is their frequency in terms of square rooted area. 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.
A small percentage of validation set could impose a set which does not distribute as the training set, resulting in worse results in test time.
However, I've still decided to stick with this split since there is no much data

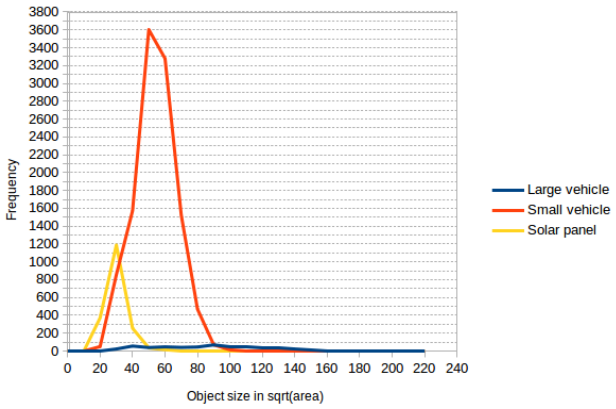

Figure 5: Dataset Objects Size Distribution

and due to the fact that I'm going to train on the full dataset anyway before submitting results (in order to slightly increase the accuracies).

4 SOTA Object Recognition Networks

Explain about SSD, YoLo, Faster RCNN, and why I chose to use Faster RCNN.

5 Challenges

1. Oriented Bounding Box
The dataset consisted of Oriented Bounding Boxes (Bounding Boxes which are not axis aligned).
The popular dataset formats (COCO, PASCAL VOC, etc) however supports axis aligned Bounding Boxes only, called Horizontal Bounding Boxes. Therefore, I had to wrap each OBB in HBB. OBB should be easier to learn because they are less strict than HBB, but due to their nature the expected score would be smaller than OBB, which better wraps the objects.

2. Classes with a point instead of Bounding Box
The utility pole class objects are marked with a point instead of 4 points Bounding Box. Therefore, for the beginning 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 but extended the point to a 1x1 bounding box, with partial success in detection accuracies.

3. Subclasses

I decided to ignore the fine-grained classification for now, since most of the detection networks modules do not support them.

4. Falsely annotated images
Some image annotations contained bounding boxes with coordiantes outside of the image. I decided to crop these coordinates to fit inside image.

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 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 requires scaling them down, since they won't fit into GPU. However, it seems that the scaling process reduces detection accuracies of classes, compared to a training done only on the standard (1000x600) resolution images. The problem can probably be solved by either increasing GPU memory or cropping the high resolution images and applying NMS in order to properly filter detection duplications.

6 Experiments Results

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.
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.

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 not metioned.

I also did not see reasons to collect training plots regarding loss and accuracies until I started optimizing them, since the first upcoming experiments have other targets.

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 the combination the scale parameter values, thus, I set the scale size to be [600, 1600] (meaning each image would be scaled twice, by the two factors). 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, but 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