Journal of Engineering and Applied Sciences
MEDWELL PUBLICATIONS

Detecting Vehicles using YOLO from Aerial Images

Shighaf Abdallah, Omar Hamdoun and Assef Jafar
Higher Institute of Applied Sciences and Technology, Damasus, Syria

Key words: Deep learning, convolutional neural networks, YOLO, COWC, VEDAI, OIRDS

Corresponding Author:
Shighaf Abdallah
Higher Institute of Applied Sciences and Technology, Damascus, Syria

Page No.: 3586-3592
Volume: 15, Issue 21, 2020
ISSN: 1816-949x
Journal of Engineering and Applied Sciences

Abstract: Detection of vehicles from aerial images is a challenging subject due to the large image resolution with small targets and variant orientations. Unfortunately, there isn't any dataset large enough to be suitable for training deep models. Therefore, we recognize COWC, large aerial image dataset to use in vehicle detection. In this project, the third version of popular YOLO is modified to vastly improve its performance on aerial data. We trained on a large amount of aerial images from COWC dataset. The proposed detector was able to achieve mAP = 95% on VEDAI dataset. It outperformed SSD and R-CNN. For the OIRDS dataset, we achieved mAP = 67% without any previous training.

INTRODUCTION

Detection from aerial images is one of the most interesting fields in computer vision because it has a wide range of applications: intelligent traffic guidance systems, animal life observation, desertification surveillance, etc. However, satellite images feature high resolution with small objects in pixels. Many features and object appearance is lost during the detection process.

Recent approaches depend on deep learning techniques with Convolutional Neural Networks to get better representation of images. In this project, we apply the third version of popular YOLO for detecting vehicles in aerial images, we trained YOLO on large number of aerial data using COWC dataset, then we transferred the knowledge (trained weights) and trained again on VEDAI.

Literature reveiw: Object detection methods can be broadly categorized into: first, classical methods which involve feature extraction followed by classification. Second, modern methods based on deep learning techniques.

Most feature extraction methods used HOG, Local Binary Pattern, Scale Invariant Feature transform, etc[1, 2]. In Zhao and Nevatia[3] assumed that car is aligned with the road direction, they combined the geometric boundary of the car body and fed it into Bayesian network. These features distinguish one type of car and can't be generalized for different types.

Cao *et al.*[4] used a modified version of HOG descriptor, they first computed HOG on local cells, then trained an AdaBoost followed by SVM classifier. Their proposed framework can only detect cars oriented horizontally as the HOG descriptor is a rotation variant descriptor.

All the methods above are based on hand-crafted features, these features cannot reach an optimal balance between the discriminability and the robustness without considering the details of the real world. An alternative way is to learn features from the data automatically in a representation fashion. In recent years the deep convolutional neural network has yielded superior performance in many computer vision tasks. A lot of works which deals with problems in computer vision field have been proposed, However, detection from aerial views is studied less due to the difficulty of acquiring aerial data.

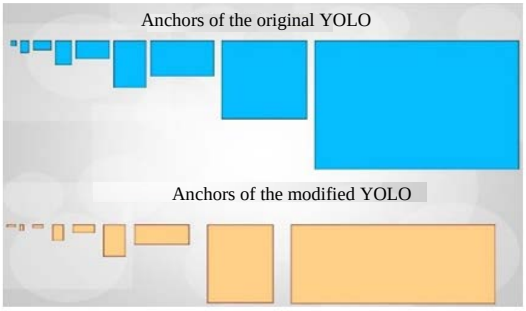

Fig. 1: The difference between anchors of the modified YOLO with the original one

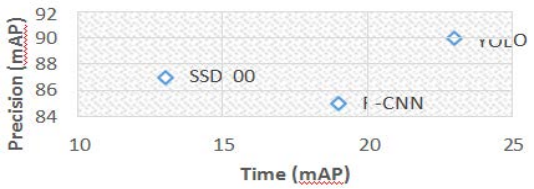

Fig. 2: Results of Videa on SSD, R-CNN, YOLO

Chen *et al*.[5] proposed a method for detecting cars in satellite images. He transformed the output of the classical DNN into several blocks of different resolutions, the proposed network has 3 feature maps, resulting in high accuracy. However, the data and images were simple with urban regions which convolve a huge number of cars and it takes 6-7 sec to process an image.

Douillard[6] tries to encounter a challenge proposed by NATO by detecting cars in satellite images. They used RetinaNet with focal loss. For the training set, they used COWC which contains very high resolution images » 13K pixels, they sliced the large images into 1000×1000 pieces with an overlap of 200 pixels. The architecture results in $f_1 = 0.91$. The sliding window technique to scan the whole image made the model very slow.

By Carlet and Abayowa[7] the second version of YOLO has been modified to detect small objects. Datasets used AFVID, AF building, VEDAI. Neovision-Helicoptor. The use of small amount of training data result in high false positive rate.

By Tang *et al*.[8] they presented an architecture based on SSD in which the bounding boxes rotate. They used a set of default boxes with various scales on each feature map. Meanwhile offsets are predicted for each default box to better match the object shape. Compared to our work, they used Vedai512 and it was able to only achieve better match the object shape. Compared to our work, they used Vedai512 and it was able to only achieve 76.26% Precision rate. While our detector was able to achieve 91% average precision (Fig. 1 and 2).

Table 1: Our model results on three different datasets

Datastes	COWC (%)	VEDAI ((%))	OIRDS (%)
AP	87.56	90.65	43.28

Table 2: Results change with input resolution

Input resolution	VEDAI (%)	OIRDS (%)	FPS (%)
256	69.58	67.59	58
416	90.65	43.28	23
608	94.67	36.98	12
832	95.11	34.54	5

Table 3: Result after changing sizes of anchor boxes with net input resolution 416

Training set	VEDAI	OIRDS
Result after modifying anchor boxes	91.82% mAP	45.33% mAP

Datasets

VEDAI: Vehicle Detection in Aerial Imagery[9]. A collection of satellite images for object detection task. The vehicles appear at different orientation and shapes with variety of backgrounds: rural, urban and desert (Table 1-3).

It has four groups: images with 1024 and 512 resolutions, RGB and infrared. All images have GSD = 12 cmPP. The images have 9 labels (plane, Boat, Camping car, Pick-up, Tractor, Truck, Van, Others). Figure 3 shows some samples, Table 4 displays some of their properties.

COWC: Cars Overhead with Context[10]. A large amount of aerial dataset, covers six areas: Toronto Canada, Selwyn New Zealand, Potsdam and Vaihingen Germany, Columbus and Utah United States. The set is designed to be difficult. It has negative examples easily mistaken for a car: boats, trailers, bushes. Columbus and Vaihingen are gray images while the rest is RGB images. The set contains >300,000 training images and 75,000 testing images. All images have GSD = 15 cmPP. This means a car would take 20-40 pixels in the image. Big trucks are not labeled while Van and Pick-ups are all labeled as cars.

The image label is one pixel at the center of a car while YOLO algorithm requires a bounding box coordinates as label in txt file for each image. We considered the width and height of 20 pixels. Figure 3 shows some samples. Table 4 displays some of their properties.

OIRDS: Overhead Imagery Research Dataset[11]. The project OIRDS constructed a set of labeled images (1,000), suitable for computer vision research. It includes 1,800 targets, each target has >30 labels describing the image and the target (image size, polygon coordinates, orientation (0-359 from top of image), % shadow, %occlusion).

Most images are of 265×265 resolution and have the property of GSD = 15 cmPP. Figure 3 shows some samples, Table 4 displays some of their properties.

Table 4: Properties for the different datasets used

Training set	Image Res	Vehicle size in image	Total number of images	GSD (cmPP)	Ratio of vehicle size to image size
VEDAI	1024×1024	40-80 pixels	1,276	12.5	0.04-0.08
COWC	256×256	24-48 pixels	≥300,000	15.0	0.09-0.18
OIRDS	256×256 to 512×512	24-48 pixels	1,000	15.0	0.09-0.18 to 0.04-0.08

Fig. 3(a-l): Samples of the different datasets used

MATERIALS AND METHODS

Deep learning models: There are three main models for detection of objects in real time based on deep learning techniques:

Faster R-CNN (Region proposals Convolutional Ceural Networks): R-CNN is built on the idea of selective search where it chooses random windows (regions) from the image called region proposals, the regions are selected by segmentation process. In each of these regions the R-CNN chooses a set of features. These features then are injected into CNN previously trained on classification. SVM with threshold is applied to recognize object existence in an image.

SSD (Single Shot Detector): SSD uses VGG-16 to extract feature-map then it detects objects using the Conv 4-3 layers. SSD does not use a delegated region proposals network, instead it resolves to very simple method, it computes both the location and class scores using small convolution filters, after extracting the feature maps, SSD applies 3×3 convolution filters for each cell to make prediction (these filters compute the results just like regular CNN filters).

YOLOV3 (You Only Look Once, Version: YOLO is a general object detector with three versions, the third version YOLOv3 has a number of improvements compared to its predecessors, it uses the new CNN feature extractor named Darknet-53, the number of predicted bounding boxes rises from 5 in the second version to 9. The most preferable enhancement is the use of three feature maps 13×13, 26×26 and 52×52. These improvements help in detecting small objects.

YOLOv3 was trained on MsCOCO and was able to achieve 28.2% mAP at 45 FPS. The architecture of YOLOv3 consists of 53 convolutional layers, most of them are 3×3 convolutions followed by maxPooling with one AvgPool and one FullyConnected layer, the scores are calculated using Softmax.

The proposed method: In order to detect small objects with aerial view, we need to train the algorithm on aerial data and modify the model structure to better detect small objects. However, the models mentioned in the previous section have pretrained weights for the MsCOCO and Pascal VOC (both contain images with side view), aerial view is rarely found.

The proposed method is at follow:

- We train the model on COWC then we transfer the trained weights and train again on VEDAI
- We modify the model to better detect small objects in an image, these modifications include: change the model's input resolutions, change anchor boxes sizes

Small object in an image which occupies a small number of pixels. Many features describe the object appearance disappear after several pooling layers and the model will not train well. Moreover, most aerial datasets contain a small number of images which makes it even harder for the convolutional layer to acquire the shape of the object.

That's why training on VEDAI at first would be impractical. At first, we need the model to acquire the representational features which describe the object completely, this means we have to train the model to detect the object before training it again to detect the same object when it's smaller (more height of the image).

In this model, we train at first on COWC dataset for the following reasons: COWC dataset contains a huge number of photos with labeled vehicles in an aerial view. It contains 300,000 images for training, it also has many negative examples which are hard to distinguish from cars.

Second, for each image in this dataset makes replicas with different orientations, this allows the algorithm to detect cars with different directions. The third, image resolution in COWC is 256×256 with 40 pixels describing the car. This means the object is considered large in the image which makes it easier to detect the representation features accurately, this in turn reduces false positive rate. After training the model on COWC, we use the resulted weights as previous weights in training the model on another dataset.

YOLOV3 modification: Yolo is not suitable for small object detection as we've mentioned earlier, many object features disappear after several maxPooling layers, YOLO needs to be modified and fintuned using appropriate data. Number of classes: YOLOv3 is designed to detect 80 different classes from MsCOCO dataset, we need only to detect one class "car", the number can be changed in the configuration file by taking into account the number of filters in the last convolutional layer.

Network resolution: YOLOv3 has an input of 416×416 pixels, this input reduces 3 feature maps 13×13, 26×26 and 52×52. VEDAI images have the resolution of 1024×1024 with GSD = 12.5 cmPP. If we consider the car has 600 cm^2 area (3 m length and 2 m width), the car will occupy 48 pixels in the image, after resize to 416, it will take 20 pixels to represent the car, meaning a vehicle may correspond to a single point on a feature map. Increasing the net resolution simply means increasing the width and height of the first layer in the net, resulting in a bigger resolution feature map and in return increasing the number of points to represent the car. For example, making the net resolution of YOLO equal to that of VEDAI will result in feature map 32×32, 64×64 and 128×128, the car will correspond to several points, this will result in good detection. However, increasing the net resolution will increase the GPU memory usage and make the network run slower.

Anchor boxes: The boxes sizes predicted in YOLO greatly affect the final result, the true positive value is calculated by the difference between these boxes and the ground truths. The boxes are calculated by k-mean clustering on a certain dataset, when dealing with aerial objects we expect most of the boxes to have small size, not squared (to fit the shape of the car).

Metrics: To evaluate the results, two metrics are used:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (1)$$

$$\text{Recall} = \frac{TP}{\text{Number\_of\_objects}} \quad (2)$$

By drawing the precision-recall curve with respect to threshold and by averaging all classes we determine the mean Average Precision (mAP).

RESULTS AND DISCUSSION

We trained YOLO on COWC datasets, the learning weight then transferred to VEDAI as previous knowledge, since, the later contains a small number of images.

COWC images are not suited for detection, they are designed for counting the number of cars in an image. The labels were just a pixel point at the center of the car, YOLO takes labels as txt file for each image with the same name, we create the labels by searching the whole image for the central point, we considered the width and height of the bounding box to be 20 pixels.

The computer used is intel® core i7, 3.40GHZ, GeForce GTX 1060 6GB. The training lasts one week with 60,000 batches using the following parameters: learning rate = 0.001, batch = 32. We used 120 K for training and 30 K for testing.

The result on Utah is AP = 87.56%. we considered one class (car), trucks and tractors were excluded, vans and pickups were considered as car. After training on COWC, we trained for 2000 batches on VEDAI using 1K for training and 300 for testing and we tested the resulted weights on OIRDS without any previous training. Table 1 displays the obtained results (Fig. 4).

Compared to Carlet and Abayowa[7], the best result was AP = 66% using YOLO. Using faster R-CNN the best result AP = 78% while our model gave AP = 90%. Our model can process 5FPS while by Carlet and Abayowa[7] it is 3FPS. The best result on VEDAI was AP = 76.26% using 512 image resolution while we used 1024 images resolution in our model.

Fig. 4(a-l): Continue

Fig. 4(a-l): Results on three datasets used: COWC, WEDAI, OIRDS

Training the algorithm on a large set on aerial data has a great effect in raising the result. When we trained YOLO on VEDAI alone, we got AP = 6.23% which is a bad result but justified. YOLO algorithm has 53 CNNs and millions of weights untrained for aerial view, training with only 1K is not sufficient for those weights to be filled properly.

Results after modifying YOLO: Table 2 displays the result on VEDAI and OIRDS for different input resolutions with frame per second rate. Increasing the net for VEDAI increases the accuracy, the result is 95.11% mAP for 832×832 in comparison to 91% mAP for 416×416.

However, it's quite the opposite for OIRDS, 34% mAP for 832×832, compared to 64% map for 2560×256. Which means increasing the net input resolution is not always better. The model would give its top accuracy when the net input resolution is equal to the test image resolution (Table 2). It's worth noting that increasing the input resolution will increase the memory usage of GPU which will result in less processing time.

Changing the anchor box would requires extracting the most frequent shapes used in the training dataset to get these shapes, we gathered the COWC dataset and the VEDAI training images, we use k-mean to calculate the sizes of the boxes that are most probabilistic to use (Fig. 1). The result is shown in Table 3.

We notice that changing the boxes size improves the result because the predicted boxes closely match the ground truth from image (a) that the model was able to detect a large number of cars, Although, the YOLO algorithm is known for its bad performance when there is a large number of objects. In image (d), the model fails to capture one of the vehicles, mistaken it for a truck.

In VEDAI dataset, the model manages to capture all the objects in the image with precise, accurate bounding boxes. But in image (h), it has mistaken the front of the truck of being a car. However, in image (e), it was able to detect the only car in the scene despite that the background contains a number of objects which resemble the shape of the car.

As for the OIRDS dataset. Which the model hadn't trained on before. The model has high detection rate, and with precise bounding boxes but in image (j), the model couldn't detect one car in the shadows because of the great change in the illumination.

We compare the results of our detector with other famous detectors (Faster R-CNN and SSD), we trained the three models (R-CNN, SSD, YOLO) on COWC for 6000 batch, we then trained on VEDAI. YOLO was the fastest algorithm to train, it only took 2000 training batches on VEDAI and achieved 91%mAP at 23FPS. While R-CNN took much time to train (4000 batches), it still slow and work on 19 FPS. However, SSD though it gives better result 87% mAP, it is far from real time process (only 13FPS).

CONCLUSION

Training a model on aerial images has some difficulties. First, aerial images seldom available in large numbers due to hardness and high cost of capturing. Second, there are multiple heights for which these images were taken, this results in different scales for the same object. Most of aerial data employed on one scale, gives poor result to other scales. In this method, we use transfer learning for training the algorithm on multiple heights by training on big number of images on large scale (COWC), then training the model again on smaller scale (VEDAI). Third, aerial images feature high resolution with small objects (small GSD), many of object features disappear during the pooling layers, fixing this by modifying input network resolution and changing anchor boxes sizes to better match the small objects. Comparing our detector with SSD and faster R-CNN showed that higher accuracy is not only achieved but also the max speed process is reached.

REFERENCES

01. Trefny, J. and J. Matas, 2010. Extended set of local binary patterns for rapid object detection. Proceedings of the Computer Vision Winter Workshop, February 3-5, 2010, Czech Pattern Recognition Society-CPRS, Prague, Czech Republic, pp: 1-7.
02. Tuermer, S., F. Kurz, P. Reinartz and U. Stilla, 2013. Airborne vehicle detection in dense urban areas using HoG features and disparity maps. IEEE. J. Sel. Top. Applied Earth Obs. Remote Sens., 6: 2327-2337.
03. Zhao, T. and R. Nevatia, 2003. Car detection in low resolution aerial images. Image Vision Comput., 21: 693-703.
04. Cao, X., C. Wu, P. Yan and X. Li, 2011. Linear SVM classification using boosting HOG features for vehicle detection in low-altitude airborne videos. Proceedings of the 2011 18th IEEE International Conference on Image Processing, September 11-14, 2011, IEEE, Brussels, Belgium, pp: 2421-2424.
05. Chen, X., S. Xiang, C.L. Liu and C.H. Pan, 2014. Vehicle detection in satellite images by hybrid deep convolutional neural networks. IEEE. Geosci. Remote Sens. Lett., 11: 1797-1801.
06. Douillard, A., 2018. Object detection with deep learning on aerial imagery. A Medium Corporation, New York, USA.
07. Carlet, J. and B. Abayowa, 2017. Fast vehicle detection in aerial imagery. Comput. Vision Pattern Recognit., Vol. 1.
08. Tang, T., S. Zhou, Z. Deng, L. Lei and H. Zou, 2017. Arbitrary-oriented vehicle detection in aerial imagery with single convolutional neural networks. Remote Sens., Vol. 9, 10.3390/rs9111170.
09. Razakarivony, S. and F. Jurie, 2015. Vehicle detection in aerial imagery: A small target detection benchmark. J. Visual Commun. Image Represent., 34: 187-203.
10. Mundhenk, T.N., G. Konjevod, W.A. Sakla and K. Boakye, 2016. A large contextual dataset for classification, detection and counting of cars with deep learning. Proceedings of the European Conference on Computer Vision, October 8-16, 2016, Springer, Amsterdam, The Netherlands, pp: 785-800.
11. Tanner, F., B. Colder, C. Pullen, D. Heagy and M. Eppolito *et al*., 2009. Overhead imagery research data set-an annotated data library & tools to aid in the development of computer vision algorithms. Proceedings of the 2009 IEEE Applied Imagery Pattern Recognition Workshop (AIPR 2009), October 14-16, 2009, IEEE, Washington, USA., pp: 1-8 .