CS230

Detect and track people aerially over Stanford campus

Priyanka Dwivedi
Department of Computer Science
Stanford University
pdwivedi@stanford.edu
Github Link: Retina Net Github

Abstract

Object Detection in Aerial Images is a challenging and interesting problem. With the cost of drones decreasing, there is a surge in amount of aerial data being generated. It is of great practical use to have models that can extract valuable information from aerial data. Retina Net is a popular single stage detector and I have retrained it on aerial images of pedestrians and bikers from the Stanford Drone Data set. I have used an existing implementation of Retina Net as a starting point and explored the effect of various parameter changes on model performance. The best model is able to get a weighted mean average precision of 0.63 on the test set. I have done detailed error analysis of the model. Finally I have fed the RetinaNet detection into a tracker than can track a pedestrian or biker across frames.

1 Introduction

Object detection in aerial photographs is an important problem due to large amount of data being generated by drones. Aerial object detection is more challenging than typical detection problems due to much smaller sizes of objects, variable lighting conditions and occlusions through trees and shadows. Feature Pyramid Networks is a structure for multi scale object detection introduced in [3]. Retina Net introduced in [4] uses a feature pyramid network that is trained using focal loss. I have trained a Retina Net on Stanford Drone Data set [1] and explored the effect of various architectural and hyper parameter changes on model performance.

The input to the model is an aerial image and the output is a list of detection. Each detection includes class name, confidence score and the coordinates of the detected bounding box. Majority of the time for the project was spent on tuning the detection model to work on Stanford Drone data set. In the last 2 weeks, I worked on integrating the trained detection model into the popular deep sort [7] algorithm for tracking detection over the video. The input for tracking is a video and the trained detection model and the output is each detection gets assigned an ID which is tracked across the video.

2 Related work

Most of the existing work in this area can be classified into two main categories: traditional approaches that rely on handcrafted features and deep learning-based approaches that rely on a convolution neural network (CNN) as feature extractor. Most of the recent works use deep learning models due to their far superior performance. CNN based object detection models can be further classified into two stage like Faster RCNN [9] that first do region proposal and single stage like YOLO [10] and RetinaNet

CS230: Deep Learning, Winter 2018, Stanford University, CA. (LateX template borrowed from NIPS 2017.)

[4] that use a single model to predict both class and bounding box coordinates. One of the recent innovations in single stage detectors has been Feature Pyramid Networks (FPN) [3] which combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections. Retina Net uses FPN with focal loss function in order to deal with data imbalance occurred by the plenty of background objects.

I have used ideas from 3 implementations of Retina Net type model on aerial object detection. [11] implements densely connected FPN on NWPU VHR-10 dataset consisting of aerial images of vehicles, bridges, ships, tennis courts etc and report state of the art results. [12] uses Retina Net on Cars Overheard data set to detect cars in high resolution aerial images. Both the above implementations detect vehicles or other building type structures that are easier to detect aerially. [8] detects pedestrians in high resolution aerial images collected over Prague and successfully shows RetinaNet can be used for detecting smaller objects and is the closest to the work done in this project. I have chosen to use Retina Net on Stanford Drone data set and did not come across any previous work that has used this data set for aerial detection.

3 Dataset and Features

For this project we used the Stanford Drone Data Set [1]. This data set consists of 60 videos shot through a UAV/Drone over 8 different locations across the Stanford campus. Each video includes annotations for 6 different classes of objects - Pedestrian, Biker, Skateboarder, Cart, Car and Bus. The Pedestrian and Biker class consist of 85% to 95% of the total annotations.

The data set is challenging for an object detection problem since each class is only a few pixels wide as shown in sample images in Figure 1. Some images also have objects occluded or under shade. These images are from hyang and deathcircle locations. The annotations show Pedestrians in pink and Bikers in red.

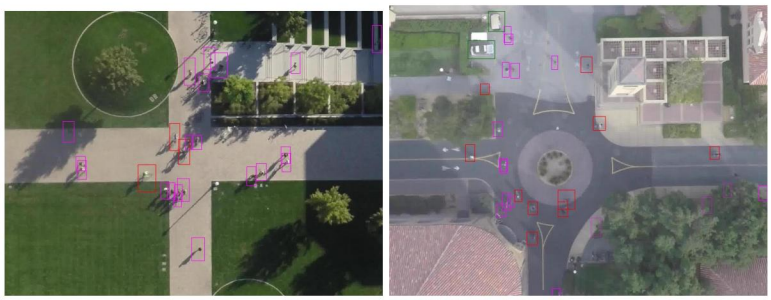

Figure 1: Stanford Drone Data Set sample images - Hyang and Death Circle locations

The training data was created by taking 12 videos across the 8 locations. Each video varies in length from 2 min to 12 minutes. 1 in every 30 frames was selected and put into the training set. This was done so we can capture frames roughly a second apart. The training set consists of 2600 test images with 35,600 annotations. Images from different videos vary in size but typical resolution is 1400x1100. Videos from the same location look different due to material changes in background. To allow the model to generalize better we chose a different set of 8 videos to be part of test/validation set. For these videos as well we sampled 1 in 30 frames. The first half of each video was used to sample frames for validation and the second half for testing. 5 % of the frames from validation and test videos were also put in training to allow the model to generalize better. Overall both validation and test set had around 600 images each with roughly 6800 annotations. No pre processing was done on the images before passing it to Retina Net.

4 Methods

4.1 RetinaNet model for object detection

Retina Net is a single stage detector that uses Feature Pyramid Network (FPN) and Focal loss for training. Convolution networks produce feature maps layer by layer and due to pooling operation the

feature maps have a natural pyramidal shape. However one problem is that there are large semantic gaps between different layers of features. The high resolution maps (earlier layers) have low-level features that harm their representational capacity for object detection. Feature pyramid network solve this by combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections. The net result is that it produces feature maps of different scale on multiple levels in the network which helps with both classifier and regressor networks.

The Focal Loss is designed to address the single-stage object detection problems with the imbalance where there is a very large number of possible background classes and just a few foreground classes. This causes training to be inefficient as most locations are easy negatives that contribute no useful signal and the massive amount of these negative examples overwhelm the training and reduces model performance. Focal loss is based on cross entropy loss as shown below and by adjusting the γ parameter, we can reduce the loss contribution from well classified examples.

$$CE(p_t) = -log(p_t)$$
$$FL(p_t) = (1 - p_t)^{\gamma} log(p_t)$$

4.2 Deep sort model for object tracking

The deep sort model introduced in [7] expands the popular SORT model used in object tracking by including in it appearance information for every detection. The appearance information is calculated by computing a 128 dimensional feature vector using a CNN.

SORT [13] is a simple framework that uses a kalman filter for tracking. While it is a good starting point, it suffers from a high number of identity switches especially in cases of occlusion or crowded scenes. This is, because the employed association metric is only accurate when state estimation uncertainty is low. The deep sort algorithm integrates into SORT, appearance information using a (CNN) that has been trained to discriminate pedestrians on a large-scale person re-identification data set. Deep sort model achieves state of the art result on MOT data set [14]. For this project I integrated the trained Retina Net model into the deep sort tracker such that detection for each frame are fed into the tracker real time as the frame is processed.

5 Experiments/Results/Discussion

5.1 Experiments on RetinaNet

For training the Retina Net model on Stanford Drone data set, I used the Keras Implementation of Retina Net in [4] as the starting point. This implementation provides pretrained weights for Resnet 50 backbone on MS COCO data set [5] and an option to train Retina Net on custom data. I ran many different experiments on the Retina Net and the main are listed below. Table 1 shows the mean average precision (MAP) on the test set for these experiments.

1. Transfer Learning - I tried using MSCOCO weights as the starting point, using a previous trained checkpoint as the starting point and training from scratch. For the very first iteration MSCOCO weights were used as a staring point. After that I saw results were much better if a previous checkpoint was used.

2. Freezing backbone - The model allows us to freeze the weights of the Resnet 50 backbone and train just the feature pyramid network. This significantly reduces training time for each epoch from 45 minutes to 18 minutes. Using a previous checkpoint to initialize weights and freeze backbone allowed me to train the model longer and better lower loss.

3. Backbone model - I experimented with 4 backbones - ResNet50, ResNet101, Mobilenet128 and DenseNet121. Pretrained MSCOCO weights are provided for only ResNet50 backbones. The other 3 models were trained from scratch. The performance on a Resnet50 backbone was much better than the other models. Training from scratch requires a lot of data as well as very careful tuning of parameters and due to lack of these, transfer learning approach worked much better.

4. Choosing smaller anchors -The RetinaNet model uses 5 default anchor boxes of size 32, 64, 128, 256 and 512. Figure 2 shows in green annotations that are captured by default anchor boxes and in red those that are missed. Substituting the 512 sized anchor box with an anchor box of size 16 allows

Experiment	Biker	Car	Bus	Pedestrian	Weighted
Resnet50 from MSCOCO weights	0.4879	0.952	0.857	0.5	0.5048
Resnet50 from MSCOCO and smaller anchors	0.486	0.936	0.789	0.7059	0.6374
Resnet50 from checkpoint, smaller anchors, frozen backbone*	0.5983	0	0.4486	0.6532	0.6328
Resnet50 from checkpoint, smaller anchors, frozen backbone, random augmentations	0.5409	0	0.5112	0.5737	0.5643
Resnet101 from scratch, small anchors	0.3013	0	0.5311	0.3244	0.3148
Mobilenet128 backbone from scratch, small anchors	0.0001	0	0	0.0003	0.0003

Table 1: Mean Average Precision from RetinaNet model. * From this exptt onwards test was changed and this test set had no cars as this is an infrequent class

us to capture most of the annotations in this data set which are smaller in size. Early experimented showed that adding the smaller anchor had a noticeable jump in mean average precision and this step was done for all subsequent experiments

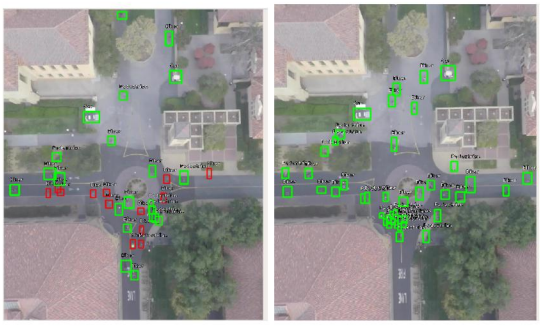

Figure 2: Left: RetinaNet with default anchor boxes, Right:Adding a small anchor box of size 16

5. Other Changes - Besides these I also tried random image augmentations, lower learning rate, change to non max suppression threshold, increase input image size etc. None of these changes had a material impact on map scores.

As the above section shows best results were obtained by using a Resnet50 backbone, initializing it with previous checkpoint, freezing backbone and using small anchors. This model results in a weighted average mean average precision of 0.63 on test set and 0.68 on validation set.

5.2 Error Analysis

I also did a detailed error analysis of the model predictions. This was done by looking at all ground truth and predicted bounding boxes on the test set and classifying into 4 classes: 1. True Positive - If there was a ground truth bounding and predicted bounding box with Intersection over Union (IOU) > 0.5 and the same class 2. Class Mismatch - If there was a ground truth bounding and predicted bounding box with IOU > 0.5 but different class 3. False Positive - Predicted bounding box has no corresponding ground truth bounding box 4.False Negative - Ground truth bounding box has no corresponding predicted box

The figure 3 shows the results of error analysis on 2 images.

Error analysis of the model shows some reasons for low map - 1. The model does get confused between the biker and pedestrian class as it can be difficult to distinguish them aerially if there are no shadows, 2. Occluded and in shadow objects are also annotated in ground truth data but are almost impossible to detect and these generate false negative predictions. 3. Shadows from trees can sometimes get predicted as an object.

Figure 3: Error Analysis of the model- True Positive - Green, Class mismatch - yellow, false positive - red, false negative - blue

5.3 Tracking using deep sort

I also fed the detection into the deep sort tracker model which assigns ID to a detection and tracks in subsequent frames. An existing implementation of deep sort tracker [15] was used but it was integrated within the Retina Net code. The deep sort model worked well in tracking across frames if the detection were separated out in space but struggled with ID switching for detection very close to each other. This is because the deep features are not able to extract meaningful information from the small bounding boxes. But it was a great learning to be able to integrate the detection model in the tracker. Using deep sort model can be a starting point but more time needs to be invested into the problem to have a high accuracy tracker. The figure 4 below shows the model tracking across two different frames.

Figure 4: Results from deep sort tracker on quad video

6 Conclusion/Future Work

I trained the RetinaNet model on aerial images from Stanford Drone data set. The experiments done here confirm that Retina Net can be trained to perform quite well on aerial images by adding a smaller anchor box. I got the best results by using a Resnet50 backbone and using transfer learning to initialize weights. The main weakness of the current model is that it suffers from mis predictions between pedestrian and biker classes and gets confused by shadows or other smaller object on the ground. I also integrated RetinaNet into deep sort tracker which is able to track detections across frames for videos that are not very crowded. As a next step, it would be good to integrate tracking results into the detection model since tracking can provide us the speed of movement of the box which can help distinguish between a biker and pedestrian. Aerial detection of people is a tough problem and to truly generalize it would be good to combine data from multiple data sets so model learns to recognize people against a variety of backgrounds.

7 Contributions

This project was done as a single person team.

References

[1] A. Robicquet, A. Sadeghian, A. Alahi, S. Savarese, Learning Social Etiquette: Human Trajectory Prediction In Crowded Scenes in European Conference on Computer Vision (ECCV), 2016.

[2] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, Piotr Dollár, Focal Loss for Dense Object Detection, arXiv:1708.02002

[3] Lin, T.-Y.; Dollár, P.; et al. Feature Pyramid Networks for Object De- tection. ArXiv e-prints, Dec. 2016, 1612.03144.

[4] Keras RetinaNet Model - https://github.com/fizyr/keras-retinanet

[5] MS COCO Data Set - http://cocodataset.org

[7] Nicolai Wojke, Alex Bewley, Dietrich Paulus. Simple Online and Realtime Tracking with a Deep Association Metric. arXiv:1703.07402

[8] Filip Bouška, Localization and Counting of Humans based on Satellite and Aerial Imagery

[9] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. ArXiv e-prints, Jun. 2015, arXiv:1506.01497

[10] Redmon J., Divvala S., Girshick R., Farhadi A. You only look once: Unified, real-time object detection; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Las Vegas, NV, USA. 27–30 June 2016.

[11] Hilal Tayara1 and Kil To Chong2, Object Detection in Very High-Resolution Aerial Images Using One-Stage Densely Connected Feature Pyramid Network

[12] Arthur Douillard, https://medium.com/data-from-the-trenches/object-detection-with-deep-learning-on-aerial-imagery-2465078db8a9

[13] A. Bewley, G. Zongyuan, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in ICIP, 2016, pp. 3464–3468.

[14] L. Leal-Taixe, A. Milan, I. Reid, S. Roth, and K. Schindler, "MOTChallenge 2015: Towards a benchmark for multi-target tracking," arXiv:1504.01942 [cs], 2015.

[15] Deep Sort Tracker Implementation - https://github.com/nwojke/deep$_sort$