View from Above: Assessing SOTA Techniques to Detect/Classify Objects in Real-Time Drone Feeds

Ryan Anderson
Georgia Institute of Technology
North Ave NW, Atlanta, GA
ryan@anderson.vc

Jayson Francis
Georgia Institute of Technology
North Ave NW, Atlanta, GA
jayson@gatech.edu

Jing-Jing Li
Georgia Institute of Technology
North Ave NW, Atlanta, GA
jli46@gatech.edu

Mario Wijaya
Georgia Institute of Technology
North Ave NW, Atlanta, GA
mwijaya3@gatech.edu

Abstract

With deep learning and computation power become much more easily accessible, the demand for accurate object detection in aerial imagery is highly sought. There are many challenges for aerial imagery object detection such as occlusion and small objects. Our goal is to improve upon the current state of the art detection algorithms and provide a comprehensive analysis. The team studied the previous approaches and experimented with models(ie. Faster RCNN and Cascade RCNN), a variety of loss functions, and a stylized translation dataset to improve the performance of the model. The team obtained Average Precision (AP) of 32.9 using Cascade RCNN- ResNet50-FPN which is much better than our baseline model with AP of 24.3 using Faster RCNN-ResNet50-FPN. The results demonstrate that there are still many ways to improve the model and to approach the problem. You can find our published code and demo video at https://github.com/gtuav/gtuav.

1. Introduction/Background/Motivation

We are trying to study the existing architecture of object detection in aerial imagery and come up with an improved approach/result in detecting the objects such as a car, pedestrian, van, truck, bus, etc.

The unique problem about the aerial imagery is that the objects are quite small when taken from the air and there is also the issue of occlusion of one object with another object that made the problem we are trying to solve much more unique than a typical object detection (ie. whether the image shown is a dog or a cat).

Currently, there are many competitions based on this topic such as the VisDrone competition and the NATO Innovation challenge to name a few. The latest VisDrone 2019 competition winner (DPNet) used the ensemble from the idea of Cascade RCNN [2], and deformable convolution [5], Feature Pyramid Network (FPN) [6].

The limit of the model is that it performed really well on the large scale object such as car but does not perform well on smaller scale object such as a pedestrian. Class imbalance and occlusion are a few of the reasons that it does not perform well for smaller scale object

Improvement in aerial object detection will impact many industries, ie. aviation, military, etc. Although the current result still not good enough for actual use, it will drive the community and future research in exploring and iterating on our idea if proven successful.

The dataset is provided by VisDrone and can be found here collected by the AISKYEYE team at Lab of Machine Learning and Data Mining, Tianjin University, China. The dataset is an aerial images captured by various drone-mounted cameras covering a wide range of aspects including location, environment (urban and country), objects (pedestrian, vehicles, etc), and density (sparse and crowded scene).

The dataset comprises of 6,471 images (training set), 548 images (validation set), and 1,610 images (testing set) with the objects annotated manually. Note that the dataset provides the ground truth of the object category where the training set and validation set were used to build the model and experimentation. The testing set is used for evaluation of our model and comparison against the benchmark from the VisDrone competition.

The dataset provided consists of an image folder with the file name that matches the file in the annotations folder with each text file includes all of the objects annotated in the image. Note that each line gives the following information as shown in Table 1.

2. Approach

2.1. High-Level

The description below provides a high-level overview of our approach to the project. The team surveyed multiple papers [8][9][13] to understand what techniques have been used for this project. After reading the papers and assessed the performance of multiple approaches such as Single State Detector (SSD) and Two-Stage Detectors, the team decided to go with Two-Stage Detector since it works better from the past competitions and approaches that people take as specified in [8] on real-world drone footage.

Customized implementation of the most promising techniques with varied architectural configurations was used for our setup. The criteria for the assessment of how well our model did is using the metric: mean Average Precision (mAP).

The team leveraged MMDetection [4] framework to enforce standardization between all of the candidate models as shown in our GitHub repository. It allows the team to experiment with many approaches within a short period amount of time where people spend months to work on the same project. Finally, the team generated the detailed profile

Name	Description
bbox_left	The x coordinate of the top-left corner of the bounding box
bbox_top	The y coordinate of the top-left corner of the object bounding box
bbox_width	The width in pixels of the object bounding box
bbox_height	The height in pixels of the object bounding box
score	1 indicates the bounding box while 0 indicates the bounding box will be ignored
object_category	Type of annotated object (ignored regions (0), pedestrian(1), people(2), bicycle(3), car(4), van(5), truck(6), tricycle(7), awning-tricycle(8), bus(9), motor(10), others(11)
truncation	Degree of object parts appears outside a frame (no truncation = 0 (truncation ratio 0%), and partial truncation = 1 (truncation ratio 1% 50%))
occlusion	Fraction of objects being occluded (no occlusion = 0 (occlusion ratio 0%), partial occlusion = 1 (occlusion ratio 1% 50%), and heavy occlusion = 2 (occlusion ratio 50% 100%))

Table 1: Dataset description

analysis of the implementation to debug and to understand why the model performs well/not good for certain object category.

2.1.1 Mean Average Precision (mAP)

Average precision (AP) is a popular metric in measuring the accuracy of object detectors like Faster R-CNN, SSD, etc. It is easier for comparison since the average precision value ranges from 0 to 1.

Using a single threshold for the IoU metric seems arbitrary and hard to compare the performance of one model against another.

mAP is one of the single metrics that would allow us to consistently compare across all models since it is derived from the precision-recall curves.

The metric mAP calculates the average precision for each class/object individually across all of the IoU thresholds. Then the metric averages the mAP for all classes to arrive at the final estimate.

2.2. Details

The team took the general models and applied them to a domain- specific that is dominated by small objects to which the past results have been struggled with. Since the problem we are trying to solve is a rapidly evolving domain, we look to improve upon past solutions and augment them with new/alternative sub-components (eg. different Region Proposal Network (RPN)) and different functions such as L1 and focal loss.

The team thought it would be successful because we are building a general model that takes pre-made State Of The Art (SOTA) sub-components to quickly experiment and understand what works and doesn't work.

There are problems such as the limited computing resources (GPU and time) and the data imbalances that we anticipated from reading the paper [8]. It is noted in the paper that every detection method from the VisDrone competition achieved inferior performance in awning-tricyle and bicycle compared to objects such as car and pedestrian due to class imbalance.

The problem that we encountered was converting the VisDrone dataset format to the Common Objects in Context (COCO) format so that it can be used with the MMDetection framework.

Another problem that we run into is designing effective experiments. Since there are many components to the model, it is hard to pinpoint the detector's subsystems to experiment with and how they are interacting with one another.

Testing a list of one-off parameters with a small number of epochs won't yield worthwhile results.

3. Model Details

All architectural configurations and experiments in this report are based on two/multi-stage detectors. This is the result of VisDrone's assessment criteria prioritizing detector accuracy over throughput. As such, our model architecture is comprised of 5 distinct subsystems: the backbone, neck, region proposal network (RPN), RoI alignment, and head components. The backbone/neck subsystem extracts features from our image that are then represented as multi-scale feature maps. The RPN processes these features to flag proposal regions containing potential objects, which are then transformed via the ROIAlign as inputs to the head subsystem. The head then classifies (or rejects) each of the ROI proposals as an object and realigns their bounding box. As a final step, our model applies non-maximal suppression (NMS) to remove duplicates from our output list of bounding boxes and labels.

Our models are built on top of the MMDetection framework, an open-source object detection tool based on PyTorch. The modular design of our architecture and the numerous MMDetection libraries allows us to augment our subsystems with SOTA techniques & rapidly experiment with modifying low-level configurations. To reproduce the work presented in this report, our team wrote several scripts for automating the training, validation, and inference activities.

3.1. Backbone & Neck Architecture

A core challenge in developing solutions for remote-viewing & aerial object detectors is the significant reduction in spatial area for objects and extreme variation in scale for objects of the same class. When evaluating aerial photography, objects may only be represented by a handful of pixels and rely on scene context to accurately classify. An approach we took to mitigate these issues was the introduction of a feature pyramid network (FPN). Our ResNet backbone takes the input image and creates a new feature map after every max pooling layer; resulting in layers of progressively less spatial detail and increasing semantic depth. The upper layers of the feature pyramid are then passed to their immediate lower pyramid in a flow that closely represents a residual. This mechanism allows us to "map" more abstract features (semantics) into our higher density spatial regions. FPN allows us to process our image at different scales and overlay context like road and crosswalk to a 20 pixel wide region where a pedestrian may be standing. In the development of our SOTA detector, we will experiment with different types of backbones (ResNet-50, ResNet-101, and ResNext) as well as a new neck alternative to FPN known as Recursive Pyramid Networks. Finally, we investigate alternatives to the vanilla 1x1 convolutions used to map the Backbone to the FPN layers. (i.e. deformable Convolutions

[5] & atrous convolutions [10]).

3.2. RPN & Head Architecture

Another challenge of aerial object detection is that extreme viewing angles often cause objects to overlap & street scenes often have localized clusters of objects. Our architecture implements a Cascade RNN, which presented significant improvement over Faster-RCNNs in reducing differences between bounding boxes and their IoU ground truth. For the aerial imagery, this had a dual benefit, because we obtain more accurate detection, but it also drives improved segmentation for scenes with crowds of people/ vehicles.

Our cascade-RCNN implements 3 sequential RoI hypotheses and bounding box updates, with increasing IoU thresholds at each sequence. This enhancement allows us to gradually refine course bounded regions; resulting in modest improvement in our IoU for detected objects. Additionally, we used the RoIAlign operator for transforming proposal regions to our head proposal. For drone imagery, ROI-Align proved vastly superior to ROI-Pooling, because its use of bilinear interpolation provides better feature map segmentation for small and distant objects. In our results, we also investigate the impact of implementing the different loss functions for classification and bounding box IoU. (See sub-section 3.2.1) Finally, our architecture introduces soft-nms (non-maximal-supression) [1] for removing overlapping ROIs. The application of soft-nms offered us slightly better performance in scenes containing a cluster of objects. This is primarily the result of soft-nms reducing the confidence weight of overlapping ROIs rather than deleting them completely. In high confidence situations; (eg people waiting at a crosswalk) we are able to retain some high confidence objects that would otherwise be lost.

3.2.1 Loss Functions

In the two-stage detectors, we used the same losses for the RPN and ROI heads. For the classification loss, we experimented with the cross entropy loss and focal loss. For the bounding box regression, we experimented with the L1 and smooth L1 losses. The cross entropy loss is defined based on the Kullback-Leibler divergence. It can be explained as the entropy of the wrong distribution of data under a true distribution.

The focal loss is primarily used in one stage detectors to balance positive and negative samples. Two-stage detectors are often trained with the cross entropy loss, and address the class imbalance problem through the two cascade stages and biased minibatch sampling.

The first stage is an object proposal mechanism that selects close to true object locations and removes the vast majority of easy negatives. The second stage classifies objects using mini batches with a predefined ratio of positive to negative examples that balance the classes. The focal loss addresses these mechanisms in one stage detectors directly via the loss function. It added a modulating factor to the cross entropy loss to down weight easy examples and focus on hard negatives. For more details on how the focal loss works, refer to the paper [7].

The smooth L1 loss combines the L1 and L2 losses with an inflection point, at which L1 loss is transitioned to L2 loss. It is a robust L1 loss that is less sensitive to outliers than the L2 loss. It eliminates the sensitivity of L2 loss to exploding gradients with certain learning rates when the regression target is unbounded.

4. Experiments and Results

4.1. Dataset

With any machine learning/deep learning project, exploratory data analysis is vital to understand the data and sometimes can be useful when thinking about what model can be helpful. The Figure 1 shows the distribution of object category for the training set and validation set respectively.

As seen from the object category distribution, it seems that the data suffer from an imbalance dataset. The performance of the recent VisDrone competition winner on minority class (object category) was not great as well and the class imbalance was the primary culprit.

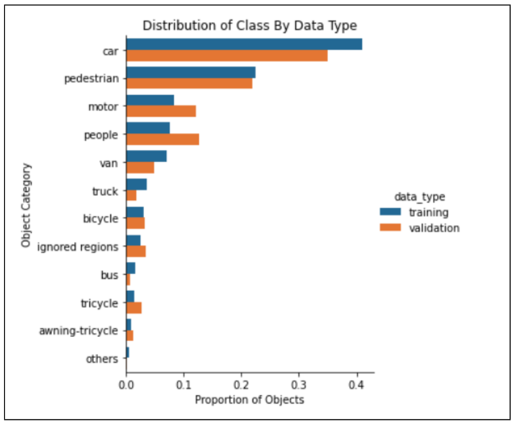

Figure 1: Distribution of objects category in the training and validation set

Another interesting thing to look at is the distribution of the number of object contains per image. We are interested in seeing whether the training set and validation set have a similar distribution.

Notice from figure 2, we can see that the distribution of the number of categories contained per image is slightly different where the validation set has several images that contain over 250+ objects.

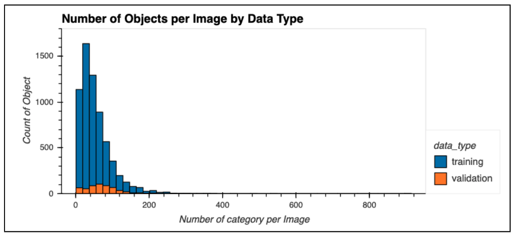

Figure 2: Distribution of objects category per image in the training and validation set

We are interested in understanding what objects appear next to each other. Instead of using hierarchical clustering which requires a lot of computation, we used the "correlation" trick to get an estimate as shown in Figure 3

Looking at figure 3 of the training set, couple of interesting things to note: people tend to appears next to motor, bicycle, and tricycle. This doesn't turn out as a surprise since the user of the motor, bicycle, and tricycle are people.

One might argue that tricycle, truck, and van should appear next to people too, however, they are enclosed space hence the "people" object are not seen.

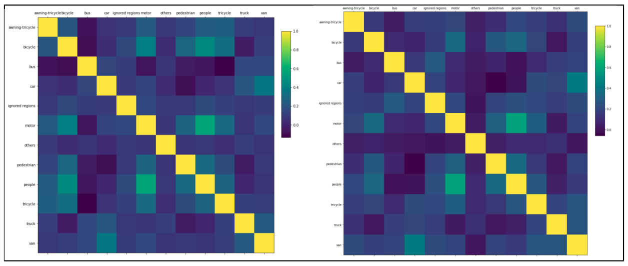

Figure 3: Correlation plot for: left (training set), right (validation set)

Lastly, we would like to validate whether the image dimension of validation is similar to the training set, otherwise, we would have to experiment with resizing the image to generate a better result.

As shown in Figure 4 for training and validation set respectively, the image dimension that is available in the validation set is similar to those in the training set.

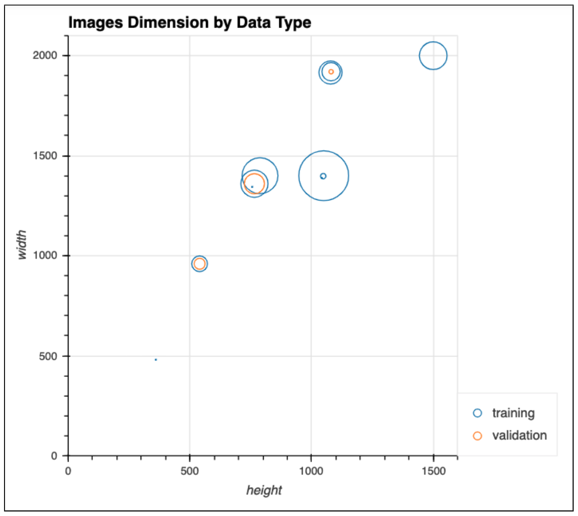

Figure 4: Distribution of Image Dimension by Data Type

4.2. Experiments

The VisDrone-DET2019 paper [8] mentioned that the performance of the last year winner did not do as well on the minority object category due to the class imbalance.

Hence, the first experiment that the team did is to grid through varieties of class losses such as focal loss (that tend to perform better as we learned from the lecture and assignment) and regression loss (bounding box) to mitigate the class imbalance problem.

Lastly, the team experiments with different models such as Faster R-CNN and Cascade R-CNN. The team used the learning curve for each model for the comparison to understand which model performs better. See Figure 5 for an example of the learning curve for the baseline detector.

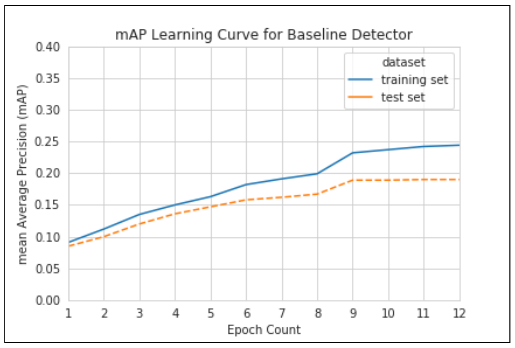

Figure 5: Baseline Detector Learning Plot

4.3. Results & Discussion

4.3.1 Comparison of Different Models

The main models that were used in the experiments are Cascade RCNN and Faster RCNN. See the paper [2] and [11] respectively for detailed information on the network architectures and how the model works.

The improvement that Cascade R-CNN has over Faster R-CNN is the iterative bounding box that uses different heads at different stages rather than the same head that is utilized over and over again. The Cascade RCNN has a specific IoU threshold from small to large for each head. The benefit of the cascaded regression for the Cascade RCNN is the resampling procedure where it provides good positive samples to the next stage that is less prone to overfitting compare to the Faster RCNN.

As shown in Table 2, we can see that the performance of average precision for CascadeRCNN is better compared to the Faster RCNN model.

Note that the team experimented with ResNet101 as a backbone but it did not converge since the performance degrades due to a small number of epochs. Had we run it for longer epochs, we would potentially observe a performance gain.

4.3.2 Comparison of Models with Class Imbalance

As shown in Table 3, the Cascade R-CNN model does not perform as well as the Faster R-CNN for smaller objects (eg. pedestrian and people) due to the Cascade-RCNN thresholding.

The issue could be with the way of Cascade-RCNN thresholding. The Cascade-RCNN pass the ROI through 3 stages where the bounding boxes are refined and the IoU threshold gradually increases.

For smaller objects, the cutoff/threshold for the final Cascade-RCNN might be higher than necessary. Had we ran more epochs, the cutoff will be more refined that will yield a better result for smaller objects.

Model	GFLOPS/FPS	# Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Faster RCNN-R50-FPN (CE/L1)	206/26	42M	24.3	39.8	25.7	15.4	36.8	38.7
Faster RCNN-R50-FPN (CE/SmoothL1)	206/26	42M	27.9	44.5	31.3	13	36.2	46.3
Faster RCNN-R50-FPN (FOCAL/L1)	206/26	42M	22.5	37.4	23.7	14.5	33.6	31.2
Faster RCNN-R101-FPN (INC)*	246/20	60M	17.9	32.4	18.0	9.5	28.3	30.3
Faster RCNN-RESNEXT50-FPN (CE/L1)	206/26	42M	—	—	—	—	—	—
CascadeRCNN-R50-RecPN (DetectoRS)	230/11	123M	20.6	33.1	21.9	10.6	33.9	39.3
CascadeRCNN-R50-FPN (CE/SmoothL1)	234/13	68M	**32.9**	**47.4**	**37.5**	17.9	**44.8**	**54.3**
DPNetV3 (A.1)	-	-	37.3	62.0	39.1	—	—	—
SMPNet (A.2)	-	-	35.9	59.5	37.4	—	—	—
DroneEye 2020 (A.3)	-	-	34.5	58.2	35.7	—	—	—
TAUN (A.4)	-	-	34.5	59.4	34.9	—	—	—

Table 2: Comparison of the model performance (classification loss/bounding box loss) - refer to the appendix of the paper [8] for DPNetV3, SMPNet, DroneEye 2020, and TAUN. *The model did not converge

Model	ped	car	tri	motor	people	van	tri	bicycle	truck	bus
FRCNN w/ CE	12.8	46.0	12.8	14.5	8.4	29.8	7.6	5.6	20.3	34.3
FRCNN w/ FOCAL	10.3	47.4	10.3	10.0	4.4	28.7	8.5	5.7	24.8	41.0
Cascade RCNN	12.0	46.1	9.7	11.5	5.4.	27.3	9.0	5.0	22.6	38.9

Table 3: Comparison of the model performance (Average Precision) by object category

4.4. Augmentation

In this section, we discuss experiments around leveraging generative networks in order to create stylized representations of our data set through style transfer. This will make two-fold contributions, 1) will enable us to augment our training data with additional examples to help generalization and 2) regularization through data by leveraging the work done here [12] in order to experiment with increasing shape bias to improve accuracy and robustness.

We make use of 7 different stylized representations from painters-by-numbers dataset. We determine a sub-optimal ratio of original images to augmented images to be 4:1 respectively. This means we increased our training data set by approximately 25%. The figure 6 is an example of the type of translation. We predict that with continuous optimization and more cycles spent on tuning parameters, we could potentially observe even better generalization performance.

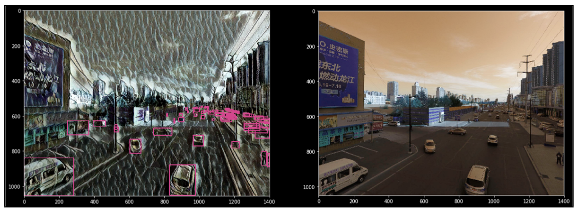
Figure 6: Visdrone stylized translation

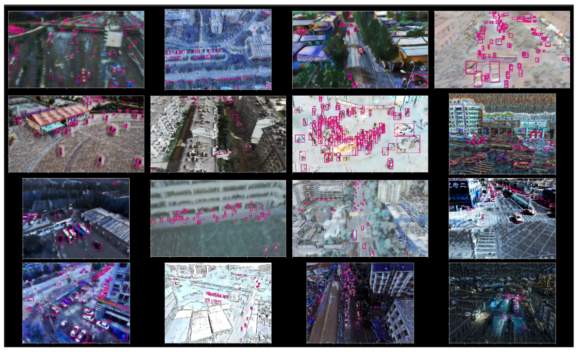
Figure 7: Visdrone stylized translation

Due to limited time and computational resources to establish a comprehensive comparison between all different models, we focused on our benchmark Cascade RCNN implementation. When training with the augmented data set, we had to make slight parameter adjustments with respect to the learning rate scheduler and the number of epochs used during training. Notice on Table 4, the augmented version of Cascade RCNN performed better for AP_{75} and AP_L.

4.4.1 Feature Pyramid Construction

The feature pyramid network (FPN) first proposed in 2016 [6] has been implemented in all the wining VisDrone Detection submissions since 2017. Its combination with ResNet to transform an image into multiple feature maps of varying channel depth and spatial resolution is well suited for this challenge. Given its ubiquity, we evaluated how variations to the FPN impacted our model performance. The first aug-

Model	GFLOPS/FPS	# Params	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
CascadeRCNN	234/13	68M	**32.9**	**47.4**	37.5	17.9	**44.8**	54.3
CascadeRCNN + Augmented	234/13	68M	31.0	39.1	**41.1**	17.9	38.8	**59.3**

Table 4: Comparison of our best model against the augmented version

mentation was replacing the backbone (used to extract features) with ResNet50, ResNet101, ResNeXt50-32x4. While R50 performed best in our ablation test we found that after much longer training windows (>>12 epochs) ResNet101 performed slightly better. Our model using ResNeXt backbone performed the best of the 3. The principal reason ResNeXt performs better is that its higher cardinality and use of group over batch normalization allows it to train better when low batch size is applied. In our experiments our batches were size 2 due to GPU memory limitations.

In 2020, a new FPN constructor was introduced: recurrent pyramid networks. Much like the name suggests, the feature map outputs of the FPN will be recursively passed 1x through the FPN's backbone. This allows key semantic features in an image to become more pronounced in the feature map. This offers SOTA improvement for foreground image segmentation but yielded less than stellar results (see DetectoRS entry in Table 2) for our problem set. This is because RecPN will contract the least prominent image feature, which also happens to be our most challenging detection objects. When we compare the DetectoRS model [10] (which implements RecPN) against our FRCNN-R50 baseline, we find it has a better mAP score when filtering on larger detection regions, but performs poorer when filtering on smaller ones.

4.4.2 Loss Functions

For the classification loss, the focal loss in Table 2 performs less than the cross entropy loss on the model Faster RCNN-R50-FPN. One of the potential reason is that focal loss dynamically scale the cross entropy loss to balance hard and easy samples, and its scaling factors need to be adjusted to achieve the best performance as mentioned in the paper [3]. For the bounding box regression, the smooth L1 loss outperforms the L1 loss probably because the smooth L1 loss combining the L1 and L2 loss is more stable than the L1 loss.

4.4.3 Overfitting

As shown in Table 3, all of the models listed works well with the bigger objects such as a car, bus, truck, and the van. However, it does not work well for smaller objects such as pedestrian, people, and bicycle.

Our model also suffers the same issue of overfitting as the competitors in VisDrone 2019 as mentioned in the paper [8]. There is an occlusion issue of the people who does not maintain a standing pose and usually involve other kinds of objects.

5. Future Work

Certainly, there are many challenges and opportunities when it comes to aerial imagery object detection.

There are many directions that one can take such as experimenting with different models, loss functions, number of epochs to train the model on, and augmentation as shown from our work in section 4.

Another interesting insight as shown in subsection 4.4 that the stylized translation of the dataset can be used to further improve the performance for several AP categories (AP_{75} and AP_L).

Several challenges that the team faced was limited computation/resource (GPU) and time. The model consists of many components and parameters hence it takes a while to train a model so the team had to think carefully about what to experiment with to yield a better result.

Within a short amount of time, the team was able to iterate on multiple experiments (models, loss functions, and augmentation) that helped us in understanding what works and does not work.

Our experiments demonstrate that current SOTA techniques optimize different parts of the VisDrone detection problem. Some architectures (Cascade-RCNN & DetectoRS) improved large feature detection, while others (R50/R101/ResNeXt FPN) returned weaker but more general results. This strongly indicates an ensemble technique will yield solutions that beat SOTA models in isolation. In future work, we will look at methods that merge ensemble predictions and implement consensus techniques for re-weighing low-confidence predictions with overlapping bounding boxes.

6. Work Division

The short summary of the delegation of work among team members is specified in Table 5. All of the team members contributed equally to write the paper. Refer to Appendix section 7.1 for the details of our work division.

Student Name	Contributed Aspects	Details
Ryan Anderson	Implementation and Hyper-parameter Tuning	Trained the model, experiment with different hyper-parameters setting, and analyze the result.
Jayson Francis	Modeling and Implementation	Convert dataset to COCO format, architecture definitions & research, and training of the model.
Jing-Jing Li	Implementation and Analysis	Studied and trained different models and loss functions and analyzed the results.
Mario Wijaya	Exploratory Data Analysis (EDA) and Analysis	EDA of the dataset used, experiment with different hyper-parameters, and analyze the result.

Table 5: Contributions of team members.

7. Appendix

7.1. Delegation of Work (Details)

- Ryan wrote scripts to streamline model training/inference generate experiment metrics for model augmentation. He worked on the baseline detector (including learning curves), and trained/tuned models with variations to their Backbone & Neck.

- Jayson worked on converting the VisDrone dataset to COCO format, researched the architecture and identified a feasible framework and viable datasets, performed augmentation with stylized translation, training of models and conducted hyper-parameter search, developed the README.md, generated video demo script and to ran inference on a demo video which has been uploaded to YouTube.

- Jing-Jing studied and trained different one-stage and two-stage models and loss functions, researched & analyzed the result specifically on the loss function

- Mario worked on understanding the dataset specifically on the distribution of the object category and relationship between each object category (correlation plot) that is in notebooks/EDA.ipynb of our GitHub repository. Researched on different models and architecture & backbone and analysis of the effect of the models with class imbalance and overfitting.

References

[1] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S. Davis. Soft-nms – improving object detection with one line of code, 2017. 3

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 1, 4

[3] Chengpeng Chen, Xinhang Song, and Shuqiang Jiang. Focal loss for region proposal network. In *Chinese Conference on Pattern Recognition and Computer Vision (PRCV)*, pages 368–380. Springer, 2018. 6

[4] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 1

[5] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017. 1, 3

[6] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 1, 5

[7] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 3

[8] Dheeraj Reddy Pailla. Visdrone-det2019: The vision meets drone object detection in image challenge results. 2019. 1, 2, 4, 5, 6

[9] George Plastiras, Christos Kyrkou, and Theocharis Theocharides. Efficient convnet-based object detection for unmanned aerial vehicles by selective tile processing. In *Proceedings of the 12th International Conference on Distributed Smart Cameras*, pages 1–6, 2018. 1

[10] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. Detectors: Detecting objects with recursive feature pyramid and switchable atrous convolution, 2020. 3, 6

[11] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 4

[12] et. al Robert Geirhos. Imagenet-trained cnns are biased towards texture; increasing shape bias improves acuracy and robustness. 2019. 5

[13] Pengfei Zhu, Longyin Wen, Dawei Du, Xiao Bian, Haibin Ling, Qinghua Hu, Qinqin Nie, Hao Cheng, Chenfeng Liu, Xiaoyu Liu, et al. Visdrone-det2018: The vision meets drone object detection in image challenge results. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018. 1