Context-based Flagging of Misclassified Objects in Aerial Imagery
Project Category: Computer Vision

Yash Chandramouli (yashc3@) [*1] **Caroline McKee (cnmckee@)** [*2] **Paul Caron (pcaron@)** [*2]

1. Abstract

Most advanced algorithms for object detection in aerial imagery analysis are still not fully accurate. Moreover, they can be fooled by adversarial examples. In this project, we investigate the possibility to improve detection algorithms utilizing the context of an image. With hand-selected contextual features, we designed various classifiers that improve the accuracy of our trained YOLOv3 model.

2. Introduction

2.1. Contributions

Yash Chandramouli worked on dataset generation and feature generating scripts. **Caroline McKee** worked on implementing the CNN to perform object detection on the dataset, the black box CNN, and comparative analyses between YOLO outputs and feature-based model outputs. **Paul Caron** worked on the various context-based prediction models. The whole team contributed equally to writing the report and general project formulation/troubleshooting.

2.2. Motivation

Recent studies in aerial imagery analyses have demonstrated the feasibility, both in the algorithmic and practical sense, of generating slightly perturbed, adversarial aerial images specifically designed to trick deep learning detection and classification algorithms into misclassifying what may appear in the image (Czaja et al., 2018). The ability to manufacture aerial images specifically designed to trick imagery classification and detection algorithms represents a huge security risk, as a hacker may utilize such methods to mount some form of digital attack. Additionally, this information could be used to design systems that evade detection/classification by aerial imagery, which could pose a threat to issues of national defense.

[*]Equal contribution [1]Department of Aeronautics and Astronautics, Stanford University [2]Institute for Computational and Mathematical Engineering, Stanford University. Correspondence to: Yash Chandramouli <yashc3@stanford.edu>, Caroline McKee <cnmckee@stanford.edu>, Paul Caron <pcaron@stanford.edu>.

Our goal for this project is to utilize the context surrounding detected objects in an aerial image to to perform a "second pass" over detected objects and flag objects which may be misclassified. Essential here is that these models, which we are calling context-based models, do not predict detection class based on the pixel values themselves, but rather hand-selected contextual features. Thus, the outputs should be invariant to any sort of tampering with pixel values (with the underlying assumption that the YOLO outputs are largely correct, which we will show to be true in our particular case). This study is framed as an exploratory analysis of which context variables are most predictive of classification, and the degree to which they can predict misclassification. While the specific imagery used for our study is aerial imagery, this study will hopefully serve as a proof of concept that learning algorithm-driven context can successfully catch misclassified detection on a variety of imagery types and modalities.

3. Related Work

There have been many other studies on similar topics. Work such as (Brubaker et al., 2014) and (Bourdev & Brandt, 2005) have demonstrated the use of cascading and boosting with simpler algorithms to adequately perform object detection tasks. In (Asha & Narasimhadhan, 2018) the authors trained an object tracking algorithm using YOLO predicted bounding boxes in order to improve the accuracy of counting vehicles from raised cameras. This demonstrated the potential to use external algorithms to enhance the classification accuracy of YOLO.

There is also much work that demonstrates the ability for hand-picked features to improve object detection results. In (Zhou et al., 2017), the authors used features to capture the difficulty of an image, such as the accuracies of simple classifiers on that image, and then used that to assign an appropriate object detection network to use for a given problem. This essentially combined multiple algorithms together and showed that non-pixel features can be used to gain meaningful insight into object classification. Similarly (Vidal-Naquet & Ullman, 2003) and (Harzallah et al., 2008) used "informative features" such as the components of a

class that most informed detection by simple networks, to improve object detection accuracy.
Perhaps the most relevant work is the work in (Divalla et al., 2009), which conducted a large-scale analysis of the importance of context-based features on object detection. They analyzed the use of a variety of context clues including 3D Geometric context, geographic information, temporally proximal images, illumination, and other aspects of an image to enhance object detection accuracy and found that the contextual reasoning increased many object detection metrics such as mAP. Our work is in the spirit of this paper, but instead of directly modifying bounding box classes with contextual information, we intend to loop contextual information into YOLO through simple classifiers. This could in theory directly inform the learning of YOLO, but for now simply serves as a secondary confirmation of the results of YOLO.

4. Dataset and Features

The dataset for this is a subset of the Large Dataset for Object Detection in Aerial Images (DOTA) (Xia et al., 2018) dataset (links). The dataset contains over 4000 images of size (6573 x 3727 pixels). Each image contains at least one of 15 classes, complete with bounding box data and image metadata. In total there are over 100,000 instances of bounding boxes in the dataset. In order to create an algorithm that is faster to train, we are using a subset of this dataset and focusing on four classes: 'plane' (class 0), 'large-vehicle' (class 1), 'ship'(class 2), and 'storage-tank' (class 3). Each image from the dataset was checked to see if it contains at least 2 occurrences of bounding boxes from that list of classes. Then each image (henceforth referred to as a "macro-image") was enhanced and split into a set of (1024 x 1024) "sub-images". This was then divided into train/dev/test sets such that the sets all have similar proportions of each class. This led to the dataset breakdown in Table 1 and Table 2 As can be seen, the distributions are relatively similar. The val and test sets are much closer with the exception of class 1 and both are relatively similar to the distribution of the training set. The reason for this discrepancy is that each sub-image contains multiple objects, so it is not always possible to find an assignment of images that yields to an even distribution of objects between sets.

	Train	Val	Test
Macro-Images	3134	385	534
Percentage	77.7	9.50	12.13.2

Table 1. Dataset Macro-image breakdown

Class	Train	Val	Test
0	5528 (15.3%)	1202 (12.7%)	1093 (12.6%)
1	8218 (22.7 %)	1386 (14.7%)	820 (9.5 %)
2	18742(51.7%)	5496 (58.1 %)	5182 (59.9 %)
3	3733 (10.3%)	1369 (14.5%)	1559 (18.0%)

Table 2. Dataset Class Breakdown

4.1. Feature Generator

In order to train algorithms to be able to classify objects based on hand-derived features, we use the knowledge that each "sub-image" in our dataset is a subsection of an associated "macro-image." Therefore we can use meta-data about the other images in the macro-image to gain insight into the likelihood of a particular set of class assignments within a sub-image. This intuitively makes sense. For example, if an object detection algorithm says there are two boats and one plane in an image, but each of the surrounding sub-images only has ships, then the scene is likely in the ocean so the plane prediction is unlikely to be accurate. Based on this concept, for each object in each image of the train set, the following features were generated:

- **Counts**- a 4-dim vector containing the counts of the 4 classes in the current image (excluding the current object). This was found both for the surrounding sub-image as well as for the larger macro-image.
- **Distance Metric**- a 4-dim vector containing a notion of the average distance to each class in the image. In order to avoid singularities where there are no objects of a certain class in an image, the metric computed was $\frac{1}{adist_i}$ where $adist_i$ is the average distance to class i in the image. This was also found for both the sub-image containing the object and for the macro-image
- **Directionality**- a 4-dim vector containing the average direction angle (in radians) to each class in the image, measured with the positive x-axis used in the image as 0 radians. This was just computed for the macro-image.

These features were combined into 6 feature schemes shown in Table 3. "Dist" refers to the distance metric,"Dir" refers to the directionality metric. Additionally, the schemes titled "Sub" only use features for the sub-image while those titled "Macro" only use features for the macro-image. The new datasets created for each feature scheme were then scaled and normalized to have 0 mean and unit variance prior to training.

5. Method

Our project utilizes two sets of machine learning algorithms:

Name	Features	Length
Sub1	Counts	4
Sub2	Counts + Dist	8
Macro1	Counts	4
Macro2	Counts + Dist	8
Macro3	Counts + Dir	8
Macro4	Counts + Dir + Dist	12

Table 3. Feature Schemes

5.1. Object Detection and Classification

The first is a convolutional neural network (CNN) trained for object detection and classification. Ultralytic's open source implementation (Ultralytics, 2018) of an object detection/classification network called YOLOv3 was trained on 77.5 percent of the dataset and trained to detect the four chosen classes.(Redmon & Farhadi, 2018). Upon training, the model achieved a mAP of 0.449 on the test data (see Figure 2). In general the YOLOv3 model struggles with objects that are small in scale. However, since the ultimate focus of this project is on improving confidence in the class of detected objects, not on training a perfect detector, a mAP of 0.449 is a sufficient baseline for our analyses. See Figure 2 for results.

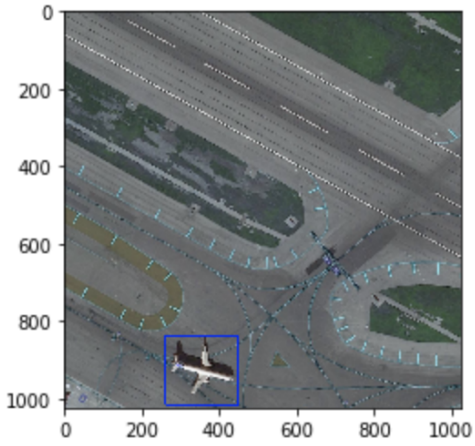

Figure 1. A sample detection of a plane using YOLOv3.

5.2. Context-Based Prediction Classifiers

In order assess which algorithm/feature scheme combination (henceforth referred to as the "context scheme") to use for the final results, a large array of classification algorithms were evaluated. Brief summaries of these algorithms and their basic working principles are below:

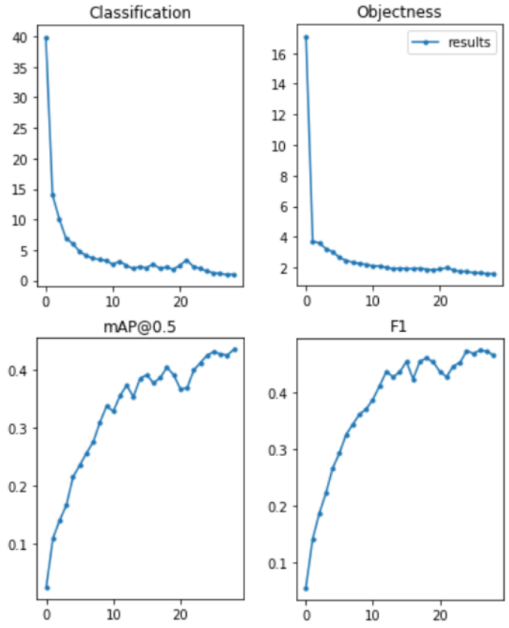

Figure 2. Performance metrics and for transfer learning-based YOLOv3 training.

- Nearest neighbors: firstly, we started using nearest-neighbor classifier. Given an integer k, $NearestNeighbor(k)$ predicts the most common class among the k training examples closest to our testing example. This classifier is useful to assess the quality of the features but is usually weakly generalizable compared to more advanced classifiers.
- Support Vector Machine: we trained two SVMs using the most common kernels, the linear kernel $K(x, y) = x^T y + c$ and the RBF kernel $K(x, y) = \exp\left(-\frac{||x-y||^2}{2\sigma^2}\right)$
- A Random Forest classifier is a powerful ensemble learning classifier that is capable of learning various types of inputs-outputs relationships and is very overfitting resistant. However, they require extensive hyperparameter tuning, especially for the maximum depth and number of splitting features.
- A 1-hidden layer perceptron with 100 hidden neurons. (Figure 3)
- AdaBoost is an ensemble learning classifier that sequentially generates classifiers $F_1, F_2, ...$ A classifier F_n is generated by optimizing with respect to θ the loss function:

$$E_n = \sum_{i=1}^{n} \mathcal{L}\left(F_{n-1}(x_i) + \alpha_n \theta(x_i), y_i\right)$$

where $\mathcal{L}$ is the global loss function of our problem.

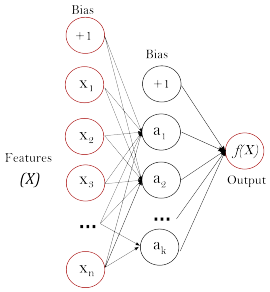

Figure 3. 1-hidden layer perceptron

	Sub1	Sub2	Macro1	Macro2	Macro3	Macro4
K-NN	0.711	0.796	0.767	0.831	0.819	0.803
LinSVM	0.824	0.892	0.852	0.882	0.857	0.883
RBF SVM	0.883	0.887	0.879	0.853	0.773	0.724
Dec. Tree	0.889	0.866	0.772	0.769	0.768	0.769
Rand. For.	0.883	0.902	0.824	0.888	0.831	0.871
MLP	0.860	0.895	0.849	0.875	0.854	0.878
Naïve Bayes	0.819	0.882	0.886	0.820	0.880	0.882
QDA	0.820	0.884	0.885	0.886	0.884	0.886
Adaboost	0.887	0.884	0.818	0.757	0.821	0.888

Figure 5. Dev set accuracy for each context scheme

- **CNN Classification of Black-box Images** A last tested algorithm involved blacking object in the image, and run the resulting image through a CNN for classification. As the CNN will not receive any pixel-based information about the object itself, this is still invariant to pixel tampering on the object itself. The specific CNN used here was the residual network ResNet50. Transfer learning was used to fine tune the network weights (pretrained on ImageNet) and completely retrain the final fully connected layer using the same training set as used for YOLOv3. The black box images, with an associated label as the class of the blacked out object, were then used as input.

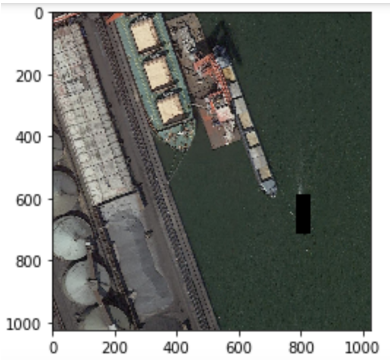

Figure 4. An example of an image with the object of interest, in this case a ship, blacked out. Images of this form, with an associated label as the class of the blacked out object, were used as input to a trained ResNet50 classification network.

6. Results and Discussion

6.1. Selecting Context-Schemes

The first task was down-selecting the context-schemes to just the ones with the highest performance. Each was evaluated on the training and dev set images with features generated using each of the 6 feature sets above. Then the best context schemes were selected by choosing the 6 with the highest dev set accuracy (highlighted in green). The resulting performance is in Figure 5. Additionally the black box classifier was run and achieved an accuracy of 0.682 on the dev set. From these results we will focus on the three best schemes: Random Forest, MLP, and Linear SVM all with Sub2.

In general the ”Sub” features outperformed the ”Macro” features. One explanation for this is that in generating the features, each object in an image generated a new feature-label pair. However, since there were not many macro-images in the dataset, this meant many of the generated features shared multiple components. For example, a macro-image with 100 ships would generate 100 feature points which have a value of 99 (pre-normalization) for the ”ship count.” This leads to the data-set not having the necessary information-richness to adequately train the algorithms, thus resulting in overfitting. It also prevented the more complex algorithms from having sufficiently information-rich data to train on, causing a bias problem as well. These problems also partially affected the sub-image features, but much less so since many more different sub-images existed in each dataset than macro-images.

Sub2 seems to have performed the best due to finding the balance between having enough features to learn more complex models, but also not having too many features that the limited dataset could not properly learn a sufficient decision boundary. The success of Random Forests is likely due to their resistance to overfitting, which minimized the effects of the repetitions in the dataset due to inherent regularization due to the random effects.

6.2. Evaluating with YOLO

To evaluate these 3 best context-schemes, first the trained YOLO algorithm was run on the test set. Then the context-schemes were trained on the ground truth training set, and then evaluated on the YOLO-produced bounding boxes, yielding the 3 confusion matrices found in Figures 6, 8, and 7. All 3 are relatively similar, with the Random Forest differing from YOLO 369 times, the Linear SVM differing 123 times, and the MLP differing 121 times. In

Random Forest w/ Sub2		Context-Scheme Prediction			
		0	1	2	3
YOLO Prediction	0	651	0	2	3
	1	29	205	9	1
	2	20	0	1203	49
	3	212	5	39	760

Figure 6. Random Forest w/ Sub2 Confusion Matrix with YOLO

Linear SVM w/ Sub2		Context-Scheme Prediction			
		0	1	2	3
YOLO Prediction	0	644	6	1	5
	1	17	216	8	3
	2	23	4	1233	12
	3	26	11	7	972

Figure 7. Linear SVM w/ Sub2 Confusion Matrix with YOLO

MLP w/ Sub2		Context-Scheme Prediction			
		0	1	2	3
YOLO Prediction	0	644	5	1	6
	1	19	212	10	3
	2	21	2	1231	18
	3	26	6	4	980

Figure 8. MLP w/ Sub2 Confusion Matrix with YOLO

particular the Random Forest tended to over-classify things as class 0. This is likely because not enough data in class 0 was present in the dataset, so the Random Forest was not able to learn decision thresholds adequately for that class. However, The other two algorithms performed similarly to YOLO, with only a few potential "flagged errors."

Nevertheless, from this alone we cannot directly tell whether these differences between YOLO outputs and our algorithm outputs were more likely due to YOLO mis-classifications or failures in the algorithms. A qualitative understanding of whether these algorithms can catch issues that YOLO cannot is found by additionally looking at the confusion matrices of the algorithms with respect to the ground truth val set. Focusing just on the Linear SVM scheme, the confusion matrix with the ground truth can be seen in Figure 9. Here one thing of note is that the linear SVM is very good at predicting ships in the ground truth data, likely since that is the largest class in the dataset and most ship appear near other ships (in ports, harbors, etc.) so they are easily distinguished by their context-features. Therefore, we can qualitatively say that there is suspicion against YOLO's prediction for any examples where the linear SVM predicts a ship and YOLO disagrees. This was then by visual inspection of multiple examples including the image in Figure 10, which was misclassified by YOLO as a large-vehicle but was correctly classified as a ship by all three context-schemes. Lastly, a test was run where the output YOLO classes were overridden by the linear SVM predicted classes whenever there was a discrepancy between the two. This led to the mAP

Linear SVM w/ Sub2		Context-Scheme Prediction			
		0	1	2	3
Ground Truth	0	644	6	1	5
	1	17	224	2	3
	2	23	4	1233	12
	3	26	15	3	972

Figure 9. Linear SVM w/ Sub2 Confusion Matrix with Ground Truth dev set

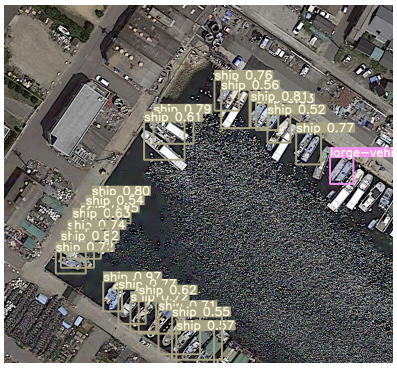

Figure 10. An example of an image with a misclassified object (the ship, which YOLOv3 predicts is a large vehicle) which was successfully flagged by our context algorithms.

score increasing to 0.451 from 0.449, indicating that the linear SVM helped rectify errors in YOLO's classification.

7. Conclusion/Future Work

Overall, this work demonstrates that context-based features can be used to train simple classifiers and can be, at least qualitatively, used as a screen to assess whether the outputs of an object detection network are reasonable or to help human data analysts be able to pick out particularly problematic images for their network. Additionally, this work showed that simple classifiers can be trained correctly for this purpose with relatively small amounts of data, meaning there is potential for this to be an easily utilizable framework for flagging mis-classified objects for most applications.

Future work will aim to directly loop these classification results into YOLO, allowing for a final update to the output bounding boxes that will improve object detection accuracy. Additionally, future work will investigate other context-based features such as average color and metadata such as weather to potentially develop more accurate context-based classifiers to supplement the primary object detection network. The Github can be found at github.com/yashc95/context4sats

References

Asha, C. and Narasimhadhan, A. Vehicle counting for traffic management system using yolo and correlation filter. In *IEEE Conference on Electronics, Computing, and Communication Technologies)*, 2018.

Bourdev, L. and Brandt, J. Robust object detection via soft cascade. volume 2, pp. 236– 243 vol. 2, 07 2005. ISBN 0-7695-2372-2. doi: 10.1109/CVPR.2005.310.

Brubaker, S. C., Wu, J., Sun, J., Mullin, M. D., and Rehg, J. M. On the design of cascades of boosted ensembles for face detection. In *ECCV*, 2014.

Czaja, W., Fendley, N., Pekala, M. J., Ratto, C., and Wang, I. Adversarial examples in remote sensing. *CoRR*, abs/1805.10997, 2018. URL http://arxiv.org/abs/1805.10997.

Divalla, S. K., Hoiem, D., Hays, J. H., Efros, A. A., and Hebert, M. An empirical study of context in object detection. In *CVPR*, 2009.

Harzallah, H., Jurie, F., and Schmid, C. Combining efficient object localization and image classification. In *LEAR INRIA Grenoble*, 2008.

links, D. https://captain-whu.github.io/dota/index.html.

Redmon, J. and Farhadi, A. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL http://arxiv.org/abs/1804.02767.

Ultralytics. yolov3, 2018. URL https://github.com/ultralytics/yolov3.

Vidal-Naquet, M. and Ullman, S. Object recognition with informative features and linear classifiers. In *The IEEE International Conference on Computer Vision (ICCV)*, 2003.

Xia, G.-S., Bai, X., Ding, J., Zhu, Z., Belongie, S., Luo, J., Datcu, M., Pelillo, M., and Zhang, L. Dota: A large-scale dataset for object detection in aerial images. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Zhou, H.-Y., Gao, B.-B., and Wu, J. Adaptive feeding: Achieving fast and accurate detections by adaptively combining object detectors. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.