Object Localization of Concentrated Animal Feeding Operations

By Minnie Ho and Jorge Troncoso

We modify an object detector based on Yolov3 to detect concentrated animal feeding operations from high-resolution satellite images. We focus on optimizing the Mean Average Precision (MAP) at an Intersection-Over-Union threshold of $\text{IOU}_t = 0.1$. We show specific techniques and hyper-parameters tuned to achieve a MAP@0.1>0.9. We compute saliency maps to visualize what the model is learning and draw conclusions from both exercises as future steps to scale our detector.

1 Introduction

Livestock production in the US has increasingly shifted to operations that raise large numbers of animals in confinement, with the largest facilities raising over 125,000 chickens or 3,000 pigs. These facilities, known as Concentrated Animal Feeding Operations (CAFOs), are estimated to produce more than 40% of US livestock and generate 335 million tons of waste per year, posing potential risks to human health and to the environment [4]. The EPA has noted that agriculture is the leading contributor of pollutants to US bodies of water and has estimated that nearly 60% of CAFOs do not hold permits [4]. In 2008, the Government Accountability Office mentioned that "no federal agency collects accurate and consistent data on the number, size, and location of CAFOs" [4]. Such data is a necessary, but currently labor-intensive and time-consuming first step towards monitoring and eventual environmental regulation of these operations.

2 Dataset and Features

Professor Daniel E. Ho at the Stanford School of Law has obtained high-resolution satellite images from the US Department of Agriculture's National Agricultural Imagery Program (NAIP), hosted on Google Earth Engine, with a maximum resolution of 15 cm per pixel [3]. Training data in the form of bounding boxes and identified CAFO types (swine, cattle, and poultry) was manually accumulated for several counties in Indiana and for one county in North Carolina [4]. The goal of our project is to build a model that can detect CAFOs, draw bounding boxes around them, and accurately classify them as poultry, swine, or cattle. We build on work from [6].

The training set is small. When summed over both Indiana and North Carolina, it consists of 2352 total CAFOs, with 1304 swine, 964 poultry, and 84 cattle CAFOs. Secondly, we observe that there is a severe class imbalance problem; Duplin County in North Carolina has no cattle CAFOs (600 swine and 300 poultry); Lagrange County in Indiana has 209 total CAFOs (13 swine, 167 poultry, and 29 cattle), and Dubois County in Indiana has 191 total (108 swine, 56 poultry, and 27 cattle). Distributions across states are clearly different, as well as across counties within a state.

CAFOs range in size from 50x30 m^2 to 600x180 m^2 (See Figure 2c). It is common to find one to three CAFOs together in a 1 km^2 area, as well as wide swaths of areas without a single CAFO. For example, in Lagrange County in Indiana, which is 1,000 km^2, CAFOs account for less than 3% of the total area of the county. Finding a CAFO is like finding the proverbial "needle in the haystack".

All CAFOs consist of metallic sheds (some long, some short) used to house the animals (See Figure 1). Cylindrical feeding tanks are sometimes visible, and (very rarely) a cow or pig can be spotted outside. CAFOs are difficult to categorize even visually in a consistent manner. For example, in North Carolina manure is stored outdoors, while in Indiana, manure is typically stored underground; hence the presence of a manure pit is not consistent enough to be useful for categorizing CAFOs. In our initial training runs, we find that CAFO sheds are sometimes mistaken for long translucent road sections, airport runways, long urban buildings or long thin parking lots. Hence, to simplify our problem, we mask urban areas and choose to detect only one class (CAFO or no CAFO) for now.

The highest possible resolution of an image from our dataset is 15 cm per pixel. However, at such high resolutions, the amount of data can be unwieldy. As an example, the images of Lagrange County at a 1 m per pixel resolution takes 4.4 GB (in Geotiff format) and 131 GB at a 15 cm per pixel resolution. Clearly, even at higher resolutions of 1 m per pixel, we need to break up the county into smaller images. We refer to this operation as "tiling", where a county is cut into consecutive, neighboring, non-overlapping RGB images of size 1024 pixels x 1024 pixels.

We note for this tiling example, the largest CAFO (600x180 m^2) easily fits in one 1024 pixel x 1024 pixel image at a resolution of 1 m per pixel. However, at a resolution of 15 cm per pixel, the largest CAFO would require 2x4 such images, and this is only if the CAFO was properly centered within the rectangle of images. For this project, we focus on 1 m per pixel resolution, especially since it is visually unclear whether higher resolutions will help improve performance.

We also mention two other relevant items of interest; unwanted regions and partial CAFOs. Google Earth Engine enables us to mask pixels outside a given region of interest by turning unwanted pixels to black. Hence, we can throw out images which have a majority of black pixels as irrelevant to our data set. In addition, as we tile the images, we will always inadvertently cut some CAFOs into pieces. We use a threshold of 0.2 to determine when a cut CAFO is "too small" and a threshold of 0.2 to determine when a CAFO has too many black pixels.

3 Methods and Related Work

There are quite a few object detection and localization algorithms, including a large list found in survey papers such as [16] and [7], as well as individual algorithms of note, such as Yolo [5], and Mask R-CNN [12]. From these, we choose Yolov3 as our initial algorithm [11] since it is simpler and faster than other methods, and achieves good performance. There are several github repositories available with Yolov3 implementations. We use the following site as the basis for our Yolov3 code [15].

Most of the previously generated weights using the Yolov3 algorithm were derived either from COCO or Pascal VOC datasets. Unfortunately, none of the 80 COCO or 20 Pascal classes remotely resemble the CAFO classes in our dataset. We chose to initialize our weights with the original Yolov3 weights which were trained on ImageNet. In addition, to overcome the limitations of our small data set, we incorporate data augmentation, since our satellite images are impervious to rotations, flips, translations, as well as color changes.

To determine the performance of our algorithm, we use the following baseline parameters in training:

- **Loss weights**: $w_{giou} = 3.31$ and $w_{obj} = 52$.
- **Thresholds**: $\text{IOU}_t = 0.5$, $\text{conf}_t = 0.01$, and $\text{nms}_t = 0.01$.
- **Learning algorithm**: optim.SGD (Pytorch) with Nesterov momentum, with the momentum value set to 0.949.
- **Learning rate**: Initial learning rate of $\lambda_0 = 0.00261$, and final learning rate of $\lambda_f = 10^{-4} * \lambda_0$, with a staircase scheduler in between.
- **Regularization**: L2 regularization of the weights, with $\lambda = 0.000489$.

We use the metrics: precision, recall, F1, Mean Average Precision, and loss, as described in the next section. We use batch normalization, with 16 samples per batch, except when we run multiclass simulations, when batch-size = 8, due to constraints on GPU memory.

To aid us in hyper-parameter tuning, we check training, validation, and test set losses against one another to guide us with directions in which to improve the model.

4 Experiments/Results/Discussion

We trained our Yolov3 algorithm on two counties: Duplin County in North Carolina, and Lagrange County in Indiana. For each county, we use only one class (the presence or absence of CAFOs). We also performed class balancing by using a ratio of 1:1 (CAFO:no CAFO) for our training and validation data.

We split our small data set using a 65/15/20 split of tiles with CAFOs, adding enough background images to form the correct ratio of background:CAFO images for the test set that we would see in practice. We augment the data using shear (-2,2), rotation (-10,10) degrees, translation (.1,.1), scale (0.9,1.1), and 50/50 LR and UP flips.

In our baseline algorithm, we retain the K-means clusters obtained by the original Yolov3 algorithm on Imagenet as our baseline.

4.1 Improving Detector Performance

For our project, we use the COCO Mean Average Precision (MAP) on the test data, as described in [8]. Most of the results quoted in this report are for an Intersection-over-Union (IOU) threshold of 0.1, denoted as MAP@0.1, since our ground-truth bounding box labels had large variation and detection is more important than precise localization in our application. To fairly depict the performance of our classifier, we draw the Precision-Recall curves for several IOU and confidence thresholds, as shown in Figure 2b for Lagrange County, using one class and our baseline configuration.

We use a loss function that is the sum of two subloss functions, which is a modification of the original Yolov3 loss function:

$$\begin{aligned} L_{loss} &= L_{giou} + L_{obj} \\ &= w_{giou}(1 - \text{GIOU}) + w_{obj}\text{nn.BCEWithLogitsLoss} \end{aligned}$$

Here, GIOU stands for the computation of the Generalized Intersection Over Union, which gives a rough "match" of detected bounding box to ground truth. The nn.BCEWithLogitsLoss stands for a Pytorch "cross-entropy" function, which gives us a classification error between the detected object and ground truth.

We plot these loss functions in Figure 2e, for the same baseline configuration. We note that while we overfit for the L_{giou}, we do not overfit so much for the L_{obj}. This shows us that we can improve performance by weighting the sub-losses in a different fashion.

We try several methods to improve MAP@0.1, as shown in Table 2a. We find that using K-means clustering (see Figure 2d) to produce anchor boxes that better match our images and checking bounding boxes to maximize recall against ground-truth (since K-means may not find a global optimum) greatly helps improve the MAP@0.1.

Lowering the importance of the bounding box accuracies, both in terms of lowering the IOU threshold and in terms of the lowering the weight w_{giou} for L_{giou}, also helps. Finally, using resized images of different resolutions (multi-class training) also improves performance, since it helps prevent over-fitting.

4.2 Saliency Maps

To visualize what the model is learning, we compute saliency maps for several images [13]. This is done by computing the gradient of the loss with respect to the input pixels. We find in Table 1 that the network trained on satellite images containing CAFOs picks up less features than a network trained on the COCO dataset [8]. We believe this is because the COCO dataset is larger and contains more object classes. The larger dataset helps prevent overfitting and the additional object classes forces the network to simultaneously learn multiple things, which allows the network to learn features that more closely resemble those that humans use to detect objects. This suggests that we may be able to extract better features to improve CAFO detection by solving a multi-object detection problem instead of a single-object detection problem. An easy way to do this, though it would increase training time and compute capacity required, would be to augment our training data with images from other satellite and drone imagery datasets, such as [10] [2] [9] [14] [1], and train the Yolo v3 network from scratch on this data. We could also try including non-satellite imagery, such as COCO images, to further increase the size of our dataset.

Another interesting finding from the saliency maps is that unlike in the classification setting, where higher pixel gradients are focused inside the object of interest, in the object detection setting, pixel gradients are usually evenly spread out over the entire image. This makes sense as small perturbations to pixels in any part of the image could cause the model to detect objects in places it shouldn't, affecting the loss value. Additionally, the edges of images usually appear darker in saliency maps generated by object detection models, meaning the derivative of the loss with respect to the edges is smaller. This shows us that the network has high confidence that edge pixels mark the end of a region of interest, and thus, no matter how much you change those pixels, the network output likely won't change.

5 Conclusion/Future Work

In this project, we confined our data to a single county and for a single class (CAFO or no CAFO). However, our MAP results on the Lagrange and Duplin counties were quite good, hence we are encouraged to use these similar techniques across counties (as well as across states). Training on more data will prevent overfitting, but will also add more variation to the type of CAFOs. Hence, it is not clear whether the test MAP will increase or drop.

To overcome the class imbalance issue, we can use FocalLoss, which weights the loss less for easily classified images (such as background) and more for the images that are more difficult to classify. In addition, our sampling of the hyperparameter space can be done far more methodically with a branch-and-search random sampling method. We anticipate such a search would greatly help when we add more data. Finally, we can also consider a more recent detector such as RetinaNet (using the Facebook Detectron package), which has been shown to incorporate the speed of Yolov3 with the performance of mask-RCNN.

6 Contributions

Minnie put particular emphasis on optimizing detector performance and establishing the data preprocessing code base. Jorge ran initial experiments presented in the milestone and later delved into the saliency map generation.

7 Acknowledgements

We thank Professor Daniel Ho, Brandon Anderson, and Ben Chugg of the Stanford Law School for giving us access to their research data/refactor code and for many helpful comments and discussions. We also acknowledge Mohamed G Mahmoud for his helpful tips on the project.

References

[1] *Cars Overhead With Context*. URL: `https://gdo152.llnl.gov/cowc/`.

[2] *DIUx xView 2018 Detection Challenge*. URL: `http://xviewdataset.org/`.

[3] *Google Earth Engine*. URL: `https://earthengine.google.com/`.

[4] Cassandra Handan-Nader and Daniel E. Ho. "Deep learning to map concentrated animal feeding operations". In: *Nature Sustainability* 2.4 (2019), pp. 298–306. DOI: `10.1038/s41893-019-0246-x`.

[5] Kaiming He et al. "Mask R-CNN". In: *2017 IEEE International Conference on Computer Vision (ICCV)* (2017). DOI: `10.1109/iccv.2017.322`.

[6] Minnie Ho and Sage Fernandez. "Object Localization of Concentrated Animal Feeding Operations". In: *CS231n Class Project* (2019).

[7] Fei-fei Li, Justin Johnson, and Serena Yeung. *Detection and Segmentation*. May 2019.

[8] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Cham: Springer International Publishing, 2014, pp. 740–755. ISBN: 978-3-319-10602-1.

[9] *Mapping Challenge*. URL: `https://www.crowdai.org/challenges/mapping-challenge`.

[10] *Object Detection in Aerial Images*. URL: `https://captain-whu.github.io/DOAI2019/dataset.html`.

[11] Joseph Redmon and Ali Farhadi. "YOLOv3: An Incremental Improvement". In: *CoRR* abs/1804.02767 (2018). arXiv: `1804.02767`. URL: `http://arxiv.org/abs/1804.02767`.

[12] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016). DOI: `10.1109/cvpr.2016.91`.

[13] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. *Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps*. 2013. arXiv: `1312.6034 [cs.CV]`.

[14] *Stanford Drone Dataset*. URL: `http://cvgl.stanford.edu/projects/uav_data/`.

[15] Ultralytics. *Ultralytics Github: Yolov3*. Nov. 2019. URL: `https://github.com/ultralytics/yolov3`.

[16] Z Zou. *Object Detection in 20 Years: A Survey*. URL: `https://www.researchgate.net/publication/333077580_Object_Detection_in_20_Years_A_Survey`.

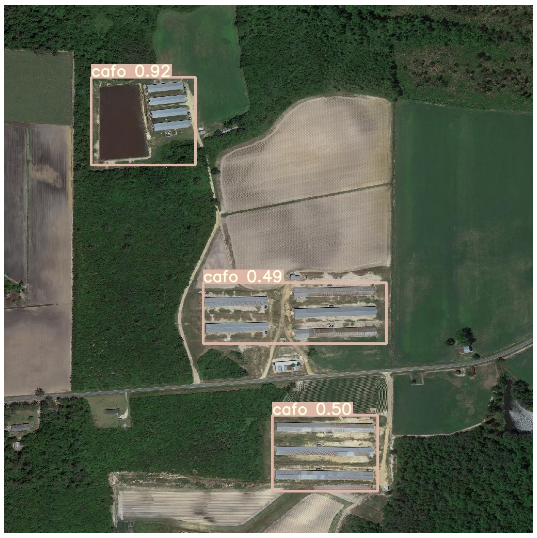


Figure 1: Example CAFO Image

Appendix

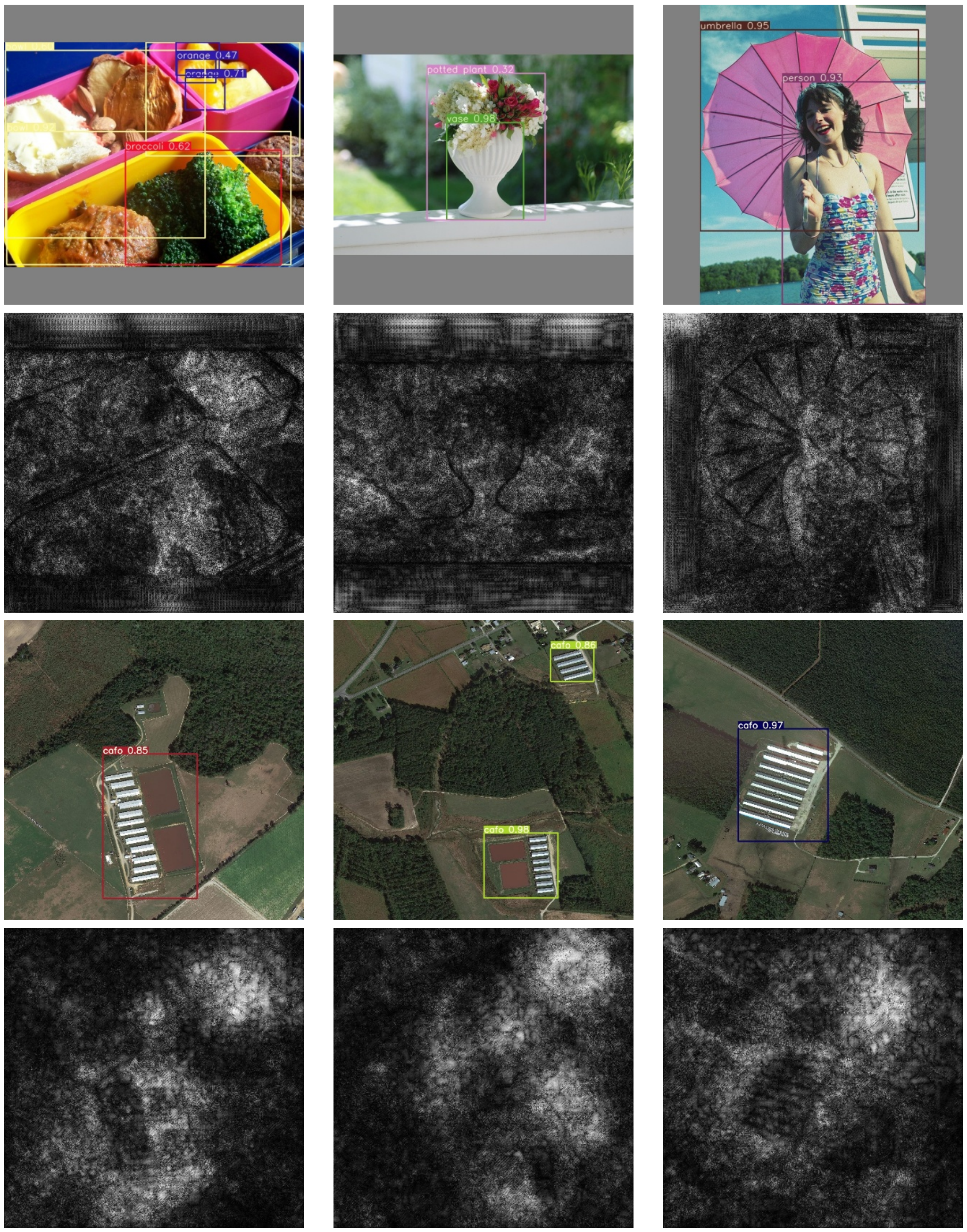


Table 1: Saliency maps and their corresponding images. Lighter pixels means greater derivative. The model trained on CAFO images picks up less features than the model trained on the COCO dataset. Edges appear darker in saliency maps generated by the COCO model.

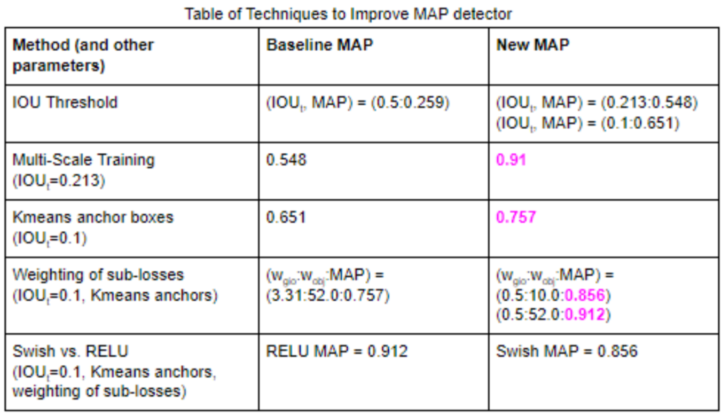
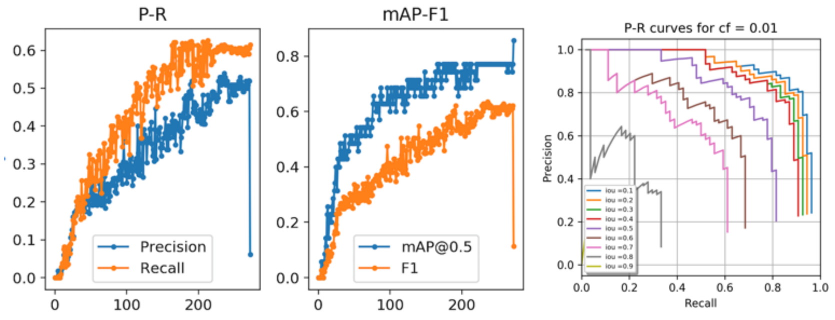


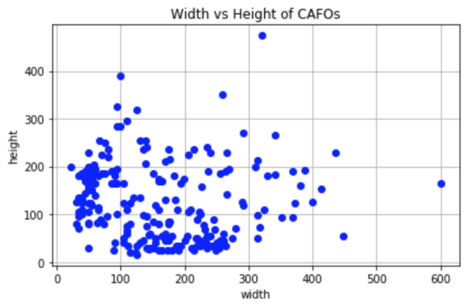
Table of Techniques to Improve MAP detector

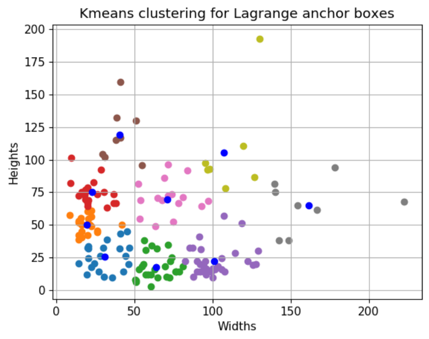
Method (and other parameters)	Baseline MAP	New MAP
IOU Threshold	$(IOU_t, MAP) = (0.5{:}0.259)$	$(IOU_t, MAP) = (0.213{:}0.548)$ $(IOU_t, MAP) = (0.1{:}0.651)$
Multi-Scale Training (IOU_t=0.213)	0.548	0.91
Kmeans anchor boxes (IOU_t=0.1)	0.651	0.757
Weighting of sub-losses (IOU_t=0.1, Kmeans anchors)	$(w_{gio}{:}w_{obj}{:}MAP)$ = (3.31:52.0:0.757)	$(w_{gio}{:}w_{obj}{:}MAP)$ = (0.5:10.0:0.856) (0.5:52.0:0.912)
Swish vs. RELU (IOU_t=0.1, Kmeans anchors, weighting of sub-losses)	RELU MAP = 0.912	Swish MAP = 0.856

(a) Table of Techniques

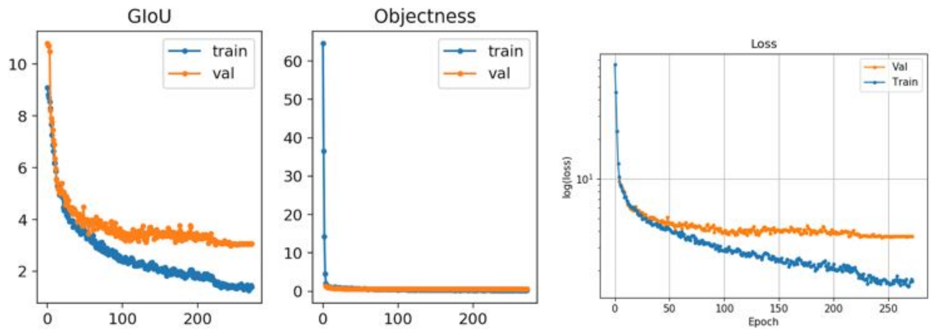


(b) (1) P-R over time, (2) MAP/F1 over time, (3) P-R over IOU_t





(c) Ground-truth boxes (d) K-means boxes



(e) SubLosses (1), (2), and Total Loss (3)