Remote Sensing Multi-class Object Counting: A YOLO Approach

Zachary Rotzal
Stanford University
Department of Computer Science
zprotzal@stanford.edu

Abstract

This paper describes current and past work on the task of multi-category object counting in remote sensing images and implements a YOLO object detection model in an attempt to solve the task. This problem poses many challenges due to varying scale across different satellite images and varying scale of objects within a single image. The YOLO method is compared to a density based counting method. The density based method outperforms the YOLO method, but it comes at a cost of higher model parameter count and the need for Near-infrared images to accompany the standard RGB image input.

1. Introduction

Accurately determining the number of items depicted in an aerial image would be beneficial to many areas of work. For example, farmers could use the technology to count the number of crops in a field. Urban planners could use the technology to count the number of cars, bicycles, and people crossing intersections throughout the day. Furthermore, disaster relief personnel could use the technology to assess damage after a storm. These uses, just a sub-sample of all the possible uses, showcase the importance and motivation for creating an accurate model to count the number of items in an aerial image.

Thus far, much of the work in counting items from satellite images, also known as remote sensing, has focused on counting a single type of object from an image. The subject of multi-class object counting in remote sensing images is nascent, with a few papers published in the beginning of 2024. The recency of these papers indicate that this topic is a growing field of study with a lot of work still to be done.

Object counting in remote sensing images is particularly challenging because of various obstacles that exist in satellite images: unknown scale, nonuniform/cluttered background, and various object orientations. Within a single image, objects have vastly different scales, ranging from large airplanes, buildings, and ships to small cars and people. Additionally, without an assumption of scale in a satellite image (e.g. 1 meter resolution), the scale between satellite images can vary greatly. Furthermore, satellite images have varied backgrounds, ranging from dense cities to farmland and oceans, all of which are captured in different lighting conditions. These complications make multi-class object counting in remote sensing images a challenging task.

Previous research developed tailored, task-specific models for object counting in the remote sensing domain. This project implements a counting model using the publicly available You Only Look Once (YOLO) object detection model and compares the results of this model with previous multi-class object counting research. The input to the YOLO model is a 1024×1024 RGB satellite image and outputs the number of detections per class. This paper uses the Northwestern Polytechnical University Multi-Object Counting (NWPU-MOC) dataset, which contains 14 unique classes and is discussed in more detail in the Dataset section 4.

2. Related Work

As [3] notes, the field of satellite image counting can broadly be divided into three approaches: density map estimation, regression, and detection methods.

The density based approach to counting was introduced by [7]. In [3] and [4], Gao et al. build on this density based approach because they found this technique to be helpful on satellite images with dense objects and large scale variation. In this setting, "dense objects" refers to many objects that are clustered together in an image. For example, a zoomed out view of a filled parking lot would be considered a dense image. [3] and [4] are also influential because they introduce two datasets for the task of object counting in remote sensing images. In [3] Gao et al. introduce the Remote Sensing Object Counting (RSOC) dataset and a method for single class object counting in satellite images. The RSOC dataset consists of 3057 images and 4 classes: ship, large-vehicle, small-vehicle, and building. While this paper helped develop the field by providing a labeled dataset and a novel density-based counting model, it

lacks the ability to count multiple categories of objects at once. Thus, [3] is often used as a baseline for other remote sensing counting research. For example, both [1] and [15] count multiple classes in a remote sensing images, using the RSOC dataset as a data source for evaluation. This difference between the dataset's original task and the objective of these models may be a source of error in computing evaluation metrics. However, this problem might be solved by the recently published [4], which introduces the NWPU-MOC dataset for the multi-class object counting task from satellite images.

Another popular approach is to use regression based methods to estimate the number of objects in a satellite image. For example, [10] implements a linear regression model and [14] implements a Gaussian mixture regression model. Furthermore, Huang et al. [5] use regression ensembles to count items in the RSOC dataset. The paper shows that the regression ensemble model performs the best when identifying and counting buildings, but is not as good as other methods when compared to identifying and counting vehicles and ships. Overall, these regression based counting techniques seem to only work well on certain objects. Specifically, the regression models tend to work well when the objects have similar scale and are evenly distributed throughout the scene.

The last technique is a detection based approach to count the number of objects in a satellite image. Faster RCNN [12], YOLO [11], and SSD [9] are popular object detectors that could be used. Once the objects in a scene are detected, counting becomes straightforward by simply summing across the class detections in the image. Gao et al. state "this solution [object detection methods] could be successful in the condition that objects are with large sizes and sparsely located but may fail in the dense cases, especially for adjoining dense buildings, densely crowded ships in the harbors and small vehicles in the parking lots" [3]. This remark seems reasonable; however, little work in the remote sensing object counting domain has been completed, so one of the aims of this paper is to analyze this claim.

3. Method

3.1. Model

This paper uses the YOLO [11] model for object detection. YOLO was chosen because the company ultralytics provides a free, public version implementation of YOLO that easily integrates into Python [6]. This easy integration and clear documentation was an important requirement so that users who are unfamiliar with the algorithm, but are still interested in counting objects in remote sensing images can quickly and easily predict on their own images. The ultralytics framework provides five YOLO tasks: detection, segmentation, classification, pose estimation, and oriented bounding boxes. The detect task was used because, in the task of counting objects from satellite images, we care about correctly identifying objects in the scene and less about the exact position and shape of the object. Additionally, the "yolov5m6u.pt" pretrained model from ultralytics was used because it took input images of 1280×1280 pixels and the NWPU-MOC dataset uses images of size 1024×1024 pixels. This model is pretrained on the COCO dataset [8]. Furthermore, this pretrained model contains approximately 41.2 million parameters, which is smaller than the chosen best model in [4] that has approximately 59.9 million parameters. The ultralytics YOLO model's maximum number of detections was increased from the default 300 to 3,600 because there is a training example consisting of 3,582 labels. This number was used as the maximum number of detections for the training, validation, and test sets.

The YOLO model uses three loss functions: box, classification, and distribution focal loss. The standard ultralytics YOLO model also uses pose and keypoint objectness loss, but these were set to zero in this project because the task of counting objects does not require knowing an object's pose. The box loss is calculated as the mean squared error of the predicted box compared to the true bounding box. The classification loss is calculated using the cross entropy loss. Lastly, the distribution focal loss is used as introduced and described in its paper [13]. The distribution focal loss is used to counter the imbalance of classes in the training data. This function computes a higher loss for less seen classes, encouraging the neural network to focus more on examples with rare objects. Furthermore, these losses were weighted by importance for the counting task in order to create an overall model loss during training. The bounding box loss was weighted by 1, the class loss by 2, and the distribution focal loss by 10. The reasoning behind these choices was that the multi-object counting task does not care so much about forming a proper bounding box, but it is important to predict the correct class for an object. For this reason, the class loss was given double the importance of the bounding box loss. The distribution focal loss was assigned the large weight of 10 because there is great class imbalance in the dataset, with some classes only appearing in tens of images and other classes appearing in over two thousand images.

3.2. Evaluation Metrics

The ultralytics YOLO model uses precision, recall, mAP50, and mAP50-95 to determine the accuracy of the model's detections. These metrics were used to assess the model's performance during training time since accurate detections are required for accurate object counting. Once training completed, the evaluation metric was updated to reflect standards in the object counting task domain. Specifically, following [4] and prior work in object counting, this paper reports final counting results using Mean Absolute

Error (MAE) and Root Mean Squared Error (RMSE) calculated for each object category separately. MAE and RMSE are defined:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y_i}|$$

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2}$$

Note that n is the number of images, y_i is the true category count, and $\hat{y_i}$ is the predicted category count from the YOLO model.

To obtain an overall score for the model, the average MAE and RMSE was computed as follows:

$$\overline{\text{MAE}} = \frac{1}{C}\sum_{i=1}^{C}\text{MAE}_i$$

$$\overline{\text{RMSE}} = \frac{1}{C}\sum_{i=1}^{C}\text{RMSE}_i$$

The variable C represents the total number of classes, which is 14 in the NWPU-MOC dataset. One aspect of this project was developing code that used the ultralytics YOLO output and NWPU-MOC dataset labels to compute these metrics.

3.3. Training

The YOLO model was trained for 100 epochs with a batch size of 1 on a NVIDIA T4 GPU. It took approximately 12 hours for the model to train. A smaller batch size was required because the maximum number of detections is large: 3,600. The YOLO model was trained with the stochastic gradient descent (SGD) optimizer with a learning rate of 0.01, momentum of 0.9, and decay of 0.0005. The first three epochs of training were used as "warm-up epochs," where the learning rate is gradually increased to its initial value in order to stabilize initial training. These hyperparameters were selected by ultralytics's training optimizer program, which uses compute resources, batch size, and other data to determine an optimizer and its most suitable hyperparameters.

4. Dataset

The NWPU-MOC dataset was used in this paper because it is one of the first datasets that labels and counts multiple objects in satellite images with various scale. The dataset was introduced in [4] with a density based approach to count the number of objects in the scene. Due to this density based approach and the requirements of ultralytics' YOLO implementation, some preprocessing of the data was required. The NWPU-MOC dataset contains pairs of RGB and Near-infrared (NIR) images. There are 3,416 pairs of images split into 2,049 training, 342 validation, and 1,025 test examples. The dataset consists of 14 classes: Airplane, Boat, Car, Container, Farmland, House, Industrial, Mansion, Pool, Stadium, Tree, Truck, Vessel, and Others. There is large class imbalance in the dataset. [4] notes that "cars appear in 2,140 images, while only 78 images contain airplanes."

Class	Density Map		YOLO	
	MAE	RMSE	MAE	RMSE
Tree	16.6072	35.3590	29.4888	70.6238
Container	1.6381	6.2787	2.0995	12.8009
Airplane	0.0615	0.4768	0.0098	0.1082
Boat	0.7884	4.8291	0.6332	3.9602
Vessel	0.1629	1.4042	0.0166	0.1951
Car	5.6273	14.4216	4.3854	16.9220
Truck	1.6133	5.1053	0.8615	4.0578
House	2.8895	6.9919	3.1883	9.0377
Industrial	0.9073	2.7487	0.7932	2.4091
Mansion	1.7766	4.9877	1.6790	4.7560
Stadium	0.1463	0.8543	0.0107	0.2399
Others	0.2107	1.3543	0.2127	1.3529
Farmland	-	-	1.9580	3.4046
Pool	-	-	1.9259	3.5002
12-Class Avg.	2.7024	7.0676	3.6149	10.5386
14-Class Avg.	-	-	3.3759	9.5263

Table 1: Counting results from the density map model in [4] and the YOLO model. Note that [4] did not list results for Farmland and Pool.

The first preprocessing step was to remove the NIR images in the dataset because the YOLO model only takes a RGB image as input. The next preprocessing step updated the labels for each image so that it conformed to the ultralytics label standards for the YOLO detection task. Specifically, the YOLO model expects one text document per image, where every line in the document represents an object with the following information: class, x center, y center, width, and height. The annotations provided in the NWPU-MOC dataset contained the class, number of objects, and (x, y) coordinates for each item in the image. These (x, y) points were used as the center coordinates for the detect task. Since the NWPU-MOC dataset currently does not have bounding box information for every object, a heuristic was used to determine an object's height and width. Ob-

Figure 1: This example of labels (top image) compared to predictions (bottom image) shows that the YOLO model can correctly predict some classes, but struggles with some dense objects like trees.

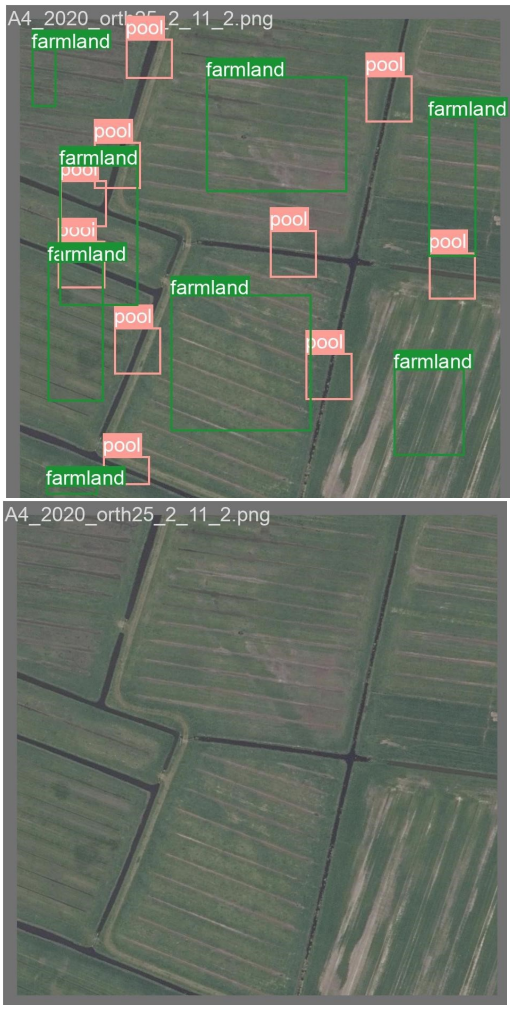

Figure 2: This example of labels (top image) compared to predictions (bottom image) shows that Farmland and Pool are difficult to predict.

jects were either determined to be large, medium, or small. Large objects were assigned a height and width of 300 pixels, medium objects a height and width of 100 pixels, and small objects a height and width of 30 pixels. The classes Farmland, Industrial, and Stadium were considered large. House, Mansion, Pool, Vessel, Container, Airplane, and Others were considered medium. Finally, the classes that made up the small category were Car, Truck, Boat, and Tree. Since there exists scale variation in the dataset, it is important to note that this heuristic is not exact and a more precise approach would be to gather exact bounding box information on every object in the images.

Lastly, the data and labels were reorganized into the directory structure that the ultralytics YOLO model expects and a data YAML file created to explain the structure to the model.

5. Results

Table 1 reports the test set results of the YOLO model trained in this paper alongside the results of the density based model presented in [4]. It is important to note that [4] did not report results for the Farmland and Pool classes, so there are two averages in the table: one for the 12 classes (not including Farmland and Pool) and one for all 14 classes.

The YOLO model performs the worst on the Tree, Car, and Container classes. The density based model also performs the worst on the Tree and Car classes. However, the density based model performs much better than the YOLO model on these classes. Since both models have trouble counting trees and cars, this indicates that these classes are particularly challenging for computer vision models to count, especially when the scene is dense.

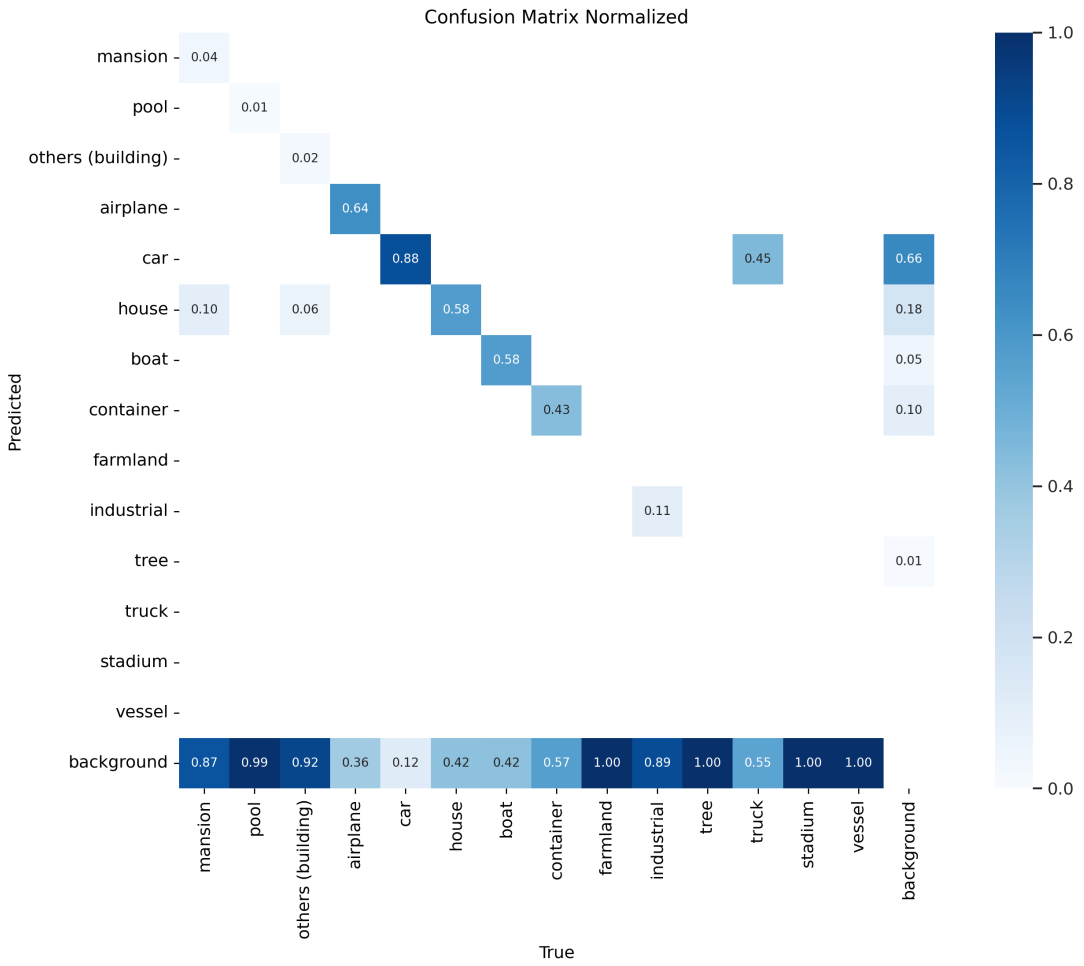

Figure 3: Confusion matrix of YOLO model on validation data.

There are a few classes where the YOLO model creates a more accurate count than the density based model. For example, the YOLO model has a lower MAE and RMSE for Airplane, Boat, and Stadium. All of these classes have a small number of examples in the training data as compared to the number of examples for Car and Tree. Therefore, this result is likely due to the extra weight this paper placed on distribution focal loss in order to counter class imbalance in the training data.

The YOLO model achieved a 14-class average MAE of 3.3759 and RMSE of 9.5263. In order to compare the YOLO model to the density based model reported in [4], a 12-class average was computed and the YOLO model achieved an overall MAE of 3.6149 and RMSE of 10.5386. Conversely, the density based model achieved an overall MAE of 2.7024 and RMSE of 7.0676. These results show that the density based model computes a more accurate count of the multi-category objects in remote sensing images; however, this better performance comes at a cost. The density based method has approximately 19 million more parameters and requires the Near-infrared image to accompany an input RGB image. The importance of this trade-off in performance will depend on a user's need for accuracy, access to NIR images, and compute resources.

Figures 1 and 2 depict some qualitative examples of the YOLO model's performance. Figure 1 is a dense scene with many trees, houses, and cars. Notice that the YOLO model struggles to identify the trees in the scene, but is able to identify many of the houses and cars. Additionally, the model incorrectly labels a bridge as another type of building and is unable to identify the body of water. Figure 2 further shows that the YOLO model struggles to detect some physically larger categories, such as Farmland and Pool. Figures 1 and 2 also shows that the NWPU-MOC dataset classifies any body of water as Pool. This is an interesting choice of label and proves difficult for computer vision models as bodies of water can be in various shapes and sizes. On the other hand, swimming pools can often fit into a rectangular

shape.

Figure 3 depicts the normalized confusion matrix of the YOLO model. One concerning aspect of the confusion matrix is that the YOLO model predicts with some regularity Car in situations that are actually Truck and Background. Another concern is that the YOLO model often incorrectly predicts background when there is an object present in the scene. These issues might be resolved through better models or additional labels and examples in the NWPU-MOC dataset.

Recall the comment in [3]: "this solution [object detection methods] could be successful in the condition that objects are with large sizes and sparsely located but may fail in the dense cases, especially for adjoining dense buildings, densely crowded ships in the harbors and small vehicles in the parking lots." One of the aims of this paper is to analyze this claim. The results above show some contrasting points. On one hand, the YOLO model could successfully identify small, dense cars, but was extremely poor at identifying dense trees. Conversely, the YOLO model could not identify large objects like farmland. Ultimately, more training data, especially data that is balanced across object categories, is needed to fully vet the claim in [3].

6. Future Work

There are two efforts that standout as possible avenues for future work. First, exact bounding boxes to the dataset labels can be added to the NWPU-MOC dataset. This added information could help train the ultralytics YOLO model on the oriented bounding boxes task, which is pretrained on DOTA, a satellite object detection dataset [2]. The added label information and a better task-aligned pretrained model may be able to obtain better results. Secondly, one could add the segmentation task to augment object detection. For example, segmentation may be able to identify the farmland and bodies of water in Figure 2. The combination of segmentation and object detection may lead to better results.

7. Conclusion

This paper explored using a YOLO model for multi-category object counting in remote sensing images of different scale. The YOLO model was compared against a density based counting model on the NWPU-MOC dataset. The YOLO model performed decently well at the task, but under performed compared the the density based model. The better performance of the density based model comes at a cost of a greater number of parameters and the need for Near-infrared images to accompany the RGB input images.

8. Contributions and Acknowledgements

This paper used ultralytics YOLO implementation, which can be found here: https://docs.ultralytics.com/.

References

[1] M. Cui, G. Ding, D. Yang, and Z. Chen. Dopnet: Dense object prediction network for multiclass object counting and localization in remote sensing images. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–15, 2024.

[2] J. Ding, N. Xue, G.-S. Xia, X. Bai, W. Yang, M. Yang, S. Belongie, J. Luo, M. Datcu, M. Pelillo, and L. Zhang. Object detection in aerial images: A large-scale benchmark and challenges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–1, 2021.

[3] G. Gao, Q. Liu, and Y. Wang. Counting from sky: A large-scale data set for remote sensing object counting and a benchmark method. *IEEE Transactions on Geoscience and Remote Sensing*, 59(5):3642–3655, May 2021.

[4] J. Gao, L. Zhao, and X. Li. Nwpu-moc: A benchmark for fine-grained multicategory object counting in aerial images. *IEEE Transactions on Geoscience and Remote Sensing*, 62:1–14, 2024.

[5] Y. Huang, Y. Jin, L. Zhang, and Y. Liu. Remote sensing object counting through regression ensembles and learning to rank. *IEEE Transactions on Geoscience and Remote Sensing*, 61:1–17, 2023.

[6] G. Jocher. Ultralytics yolov5, 2020.

[7] V. Lempitsky and A. Zisserman. Learning to count objects in images. In J. Lafferty, C. Williams, J. Shawe-Taylor, R. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010.

[8] T.-Y. Lin, M. Maire, S. Belongie, L. Bourdev, R. Girshick, J. Hays, P. Perona, D. Ramanan, C. L. Zitnick, and P. Dollár. Microsoft coco: Common objects in context, 2015.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016.

[10] N. Paragios and V. Ramesh. A mrf-based approach for real-time subway monitoring. In *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, volume 1, pages I–I, 2001.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection, 2016.

[12] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[13] L. Tao, M. Dong, and C. Xu. Dual focal loss for calibration, 2023.

[14] Y. Tian, L. Sigal, H. Badino, F. De la Torre, and Y. Liu. Latent gaussian mixture regression for human pose estimation. pages 679–690, 11 2010.

[15] W. Xu, D. Liang, Y. Zheng, J. Xie, and Z. Ma. Dilated-scale-aware category-attention convnet for multi-class object counting. *IEEE Signal Processing Letters*, 28:1570–1574, 2021.