THE FINAL PROJECT ON DRONE DATA SET USING MASK R-CNN MACHINE LEARNING CS667/767

MINGYONG LIU, EMMANUEL OBI, PAT GANESAN, DIPAK DULAL
SUBMITTED TO
DR. DA YAN
DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF ALABAMA AT BIRMINGHAM

DECEMBER 7, 2020

1. Introduction

Object detection is a demanding and challenging task in computer vision via machine learning/deep learning, which helps us understand and analyze scenes in images or videos. There are a plethora of models designed to this end, and Mask R-CNN is the one with an outstanding performance. Mask R-CNN is an extension of Faster R-CNN model that does not only detect and classify items, but also adds an output model for prediction a mask for each detected object. Mechanistically, it does not pool all Regions of Interest (RoI) as does Faster R-CNN, but instead preserves their spatial locations. This facilitates instance segmentation that cannot be achieved by Faster R-CNN. Although Mask R-CNN can be slower in making a prediction as compared to alternate models such as YOLO, but its prediction is more accurate.

In this project, we use the Mask R-CNN model on the provided drone data set and adopt a transfer training strategy by taking advantage of pre-trained COCO weight. As a result, our model is able to localize one or multiple objects in an image and classify them into their respective categories.

2. Checklist

We imported and installed the following packages for our model:

- os
- sys
- jason
- numpy
- skimage
- matplotlib
- pickle
- from mrcnn imported utils,modellib, visualize, config
- tenserflow

3. Tuning and Customization of Dataset

For general detection, the drone data set contains a total of 8040 images with annotated objects, and there is a total of five classes ("Head," "Thread," "Nut," "Washer," "Pin") plus a background class. Initially we split the data into three categories: 70% for training, 15% for validation, and 15% for testing purposes. However, we ended up using only 5% of the data set for testing due to the limited RAM we have in Google Colab.

We first loaded the pre-trained COCO weight, and then fine-tuned the model with training and validation sets under multiple conditions (see tables below). Subsequently, we selected the best model based on the validation loss, and used it for object detection and segementation from testing pictures.

During the fine-tuning step, we used the following training strategies:

Table 1. All-Layer Training Strategies

Parameters	Strategy1	Strategy2	Strategy3	Strategy4	Strategy5	Strategy6
No. of epoch for heads	20	20	20	20	20	20
Learning Rate for heads	0.001	0.001	0.001	0.001	0.001	0.001
No. of epoch for all	40	40	60	60	80	80
Learning Rate for all	0.0001	0.00001	0.0001	0.00001	0.0001	0.00001

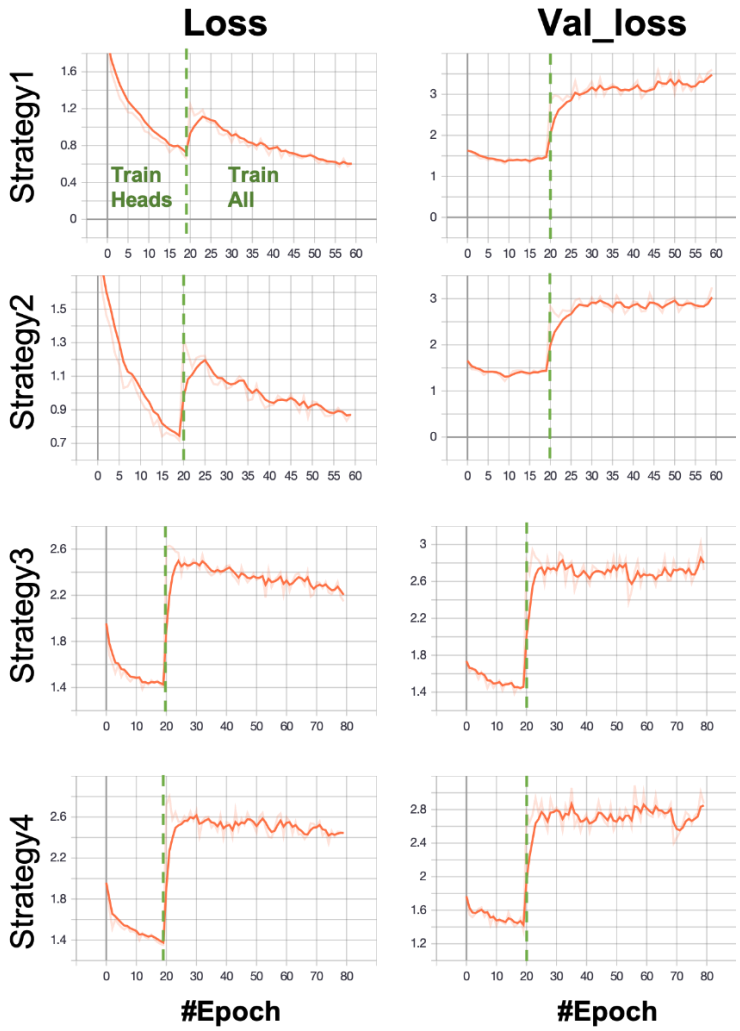


Figure 1. The convergence graph of four out of six strategies. The head layer was trained for 20 epochs, and then all layers were trained following the strategies 1-4 indicated in Table 1. Convergence of a model was determined based on the loss and val_loss. As shown in the figure, the loss increases when all-layer training is performed, which makes the model hard to converge.

Table 2. Head-only Training Strategies

Parameters	Strategy1	Strategy2	Strategy3
No. of epoch for head	40	40	60
Learning Rate for heads	0.001	0.0001	0.001

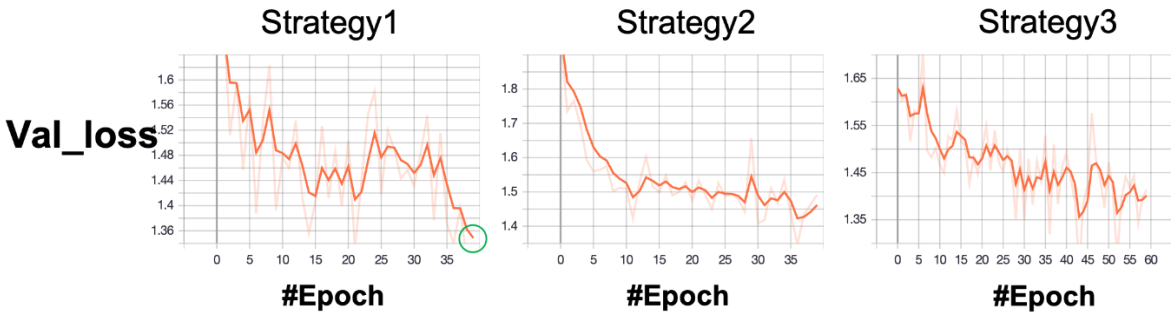


Figure 2. Val_loss for different head-only training strategies. The head layer was trained with the three strategies indicated in Table 2. Convergence of a model was determined based on the loss and val_loss. Circled is the val_loss of the model that was selected for object detection.

Average Precision

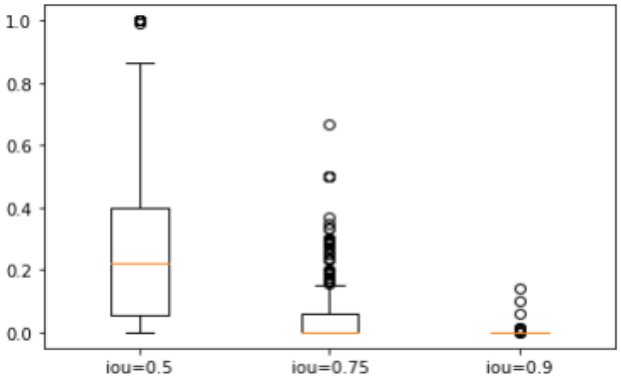
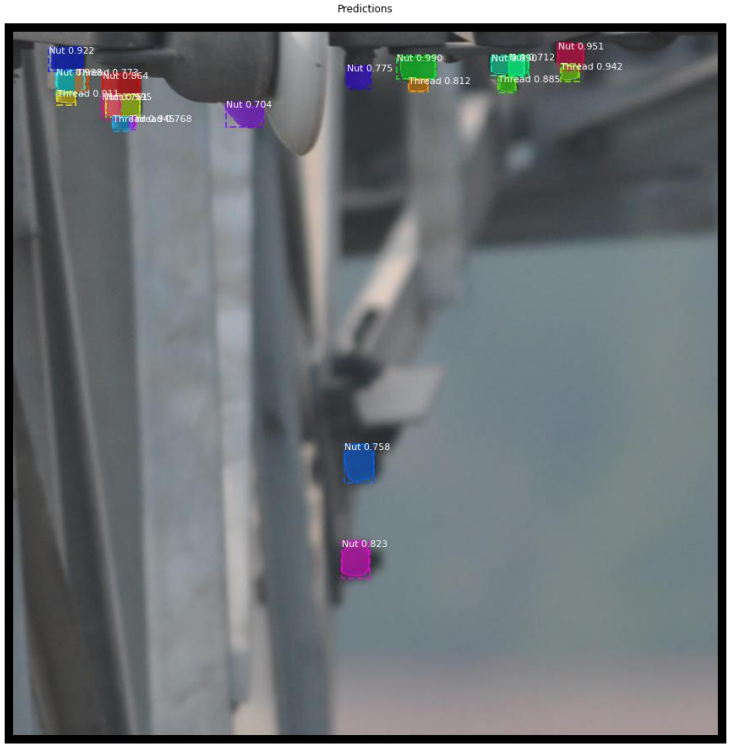


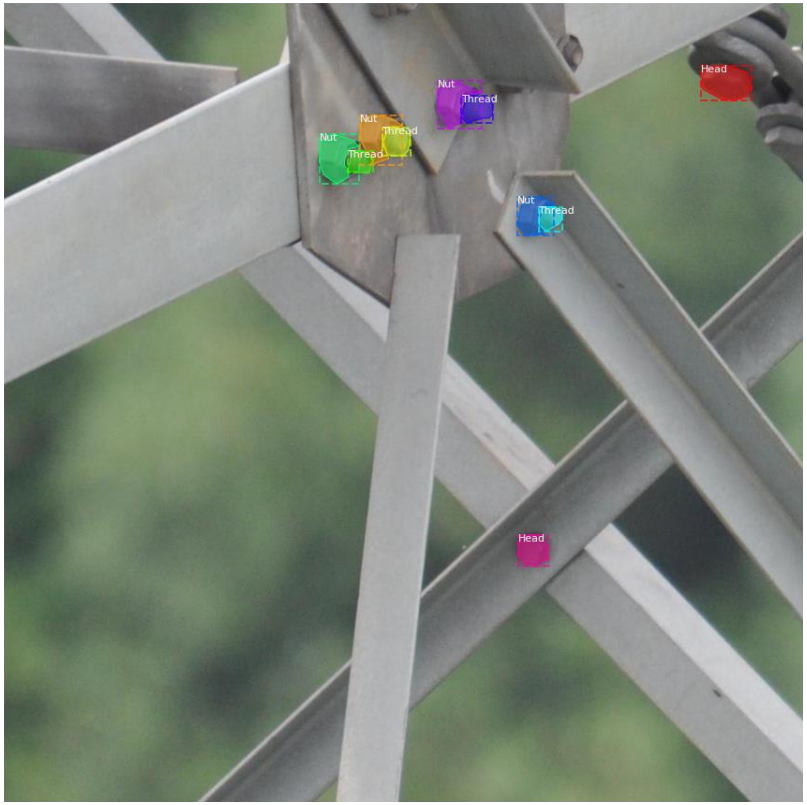
Figure 3. mAP with three different thresholds. An average precision score was calculated for all the classes in each testing picture with the indicated Intersection over Union (IoU) threshold. The summary of these scores for all the testing pictures was visualized using a box plot.

4. Top Ten Detection Result

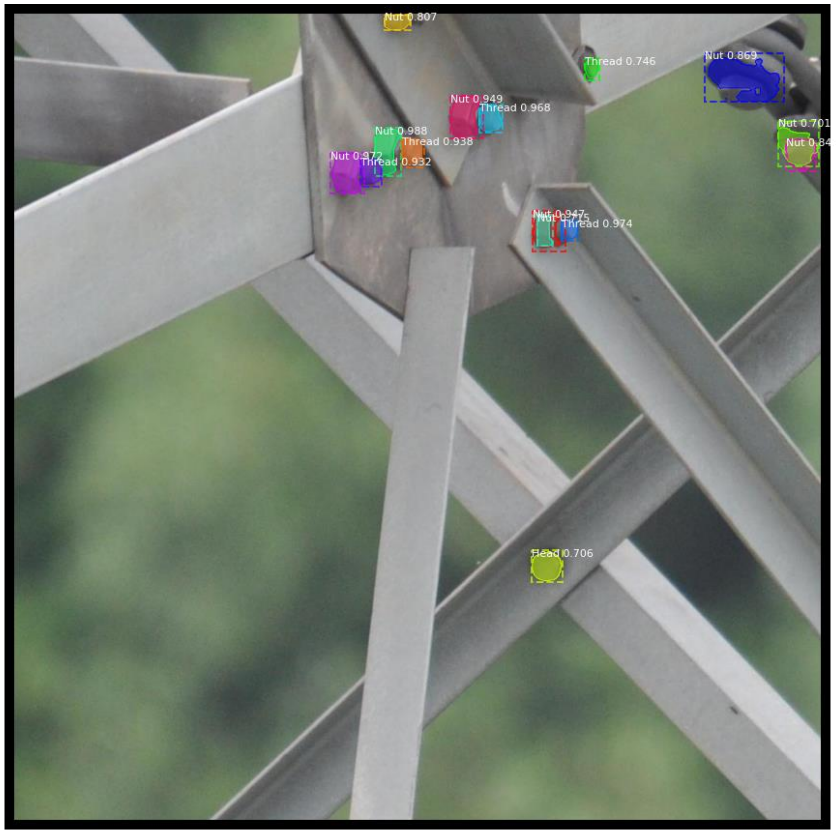
Following are the ten best-detected results. The pictures on the left column are the ground truth, and the pictures on the right column are predicted. As a result, our model can successfully detect and classify small objects. In some cases, it can also identify items that are missed by human annotation.

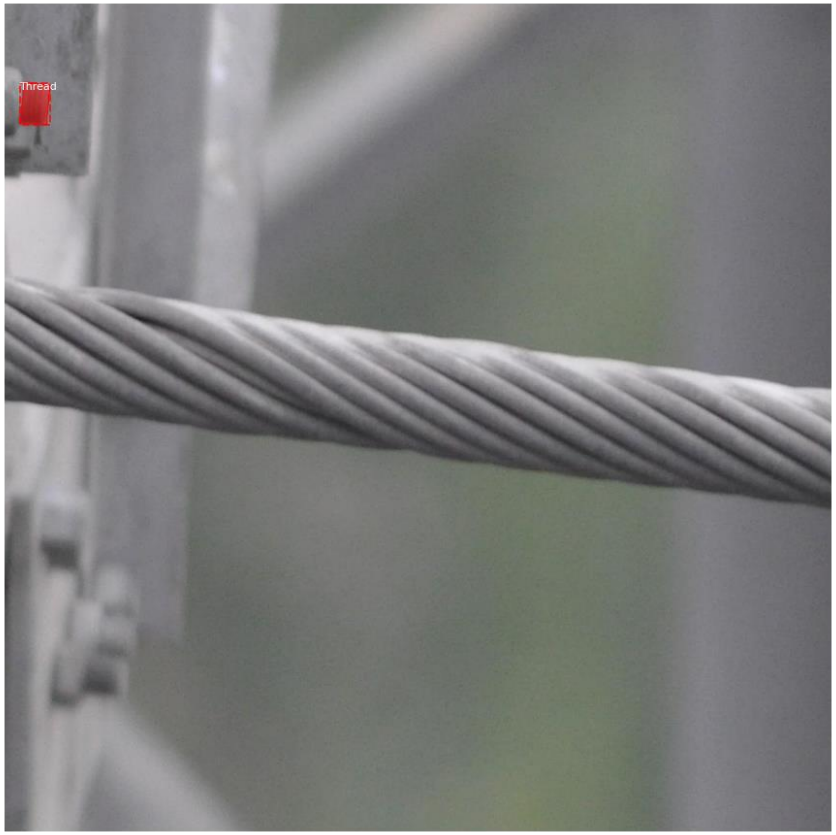
Ground Truth	Predicted
	

Predictions

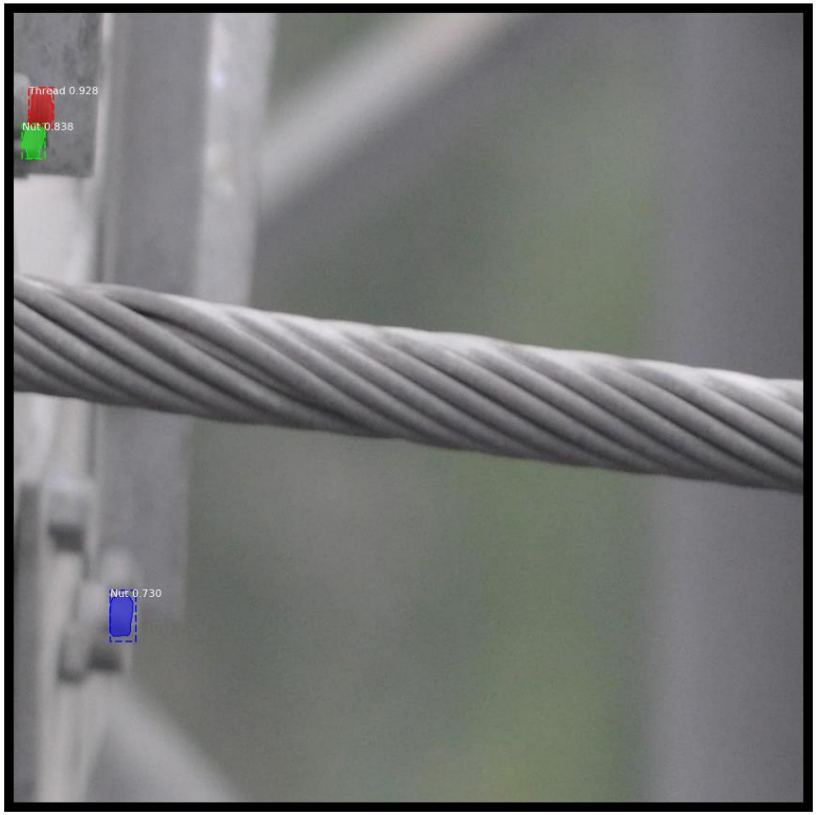


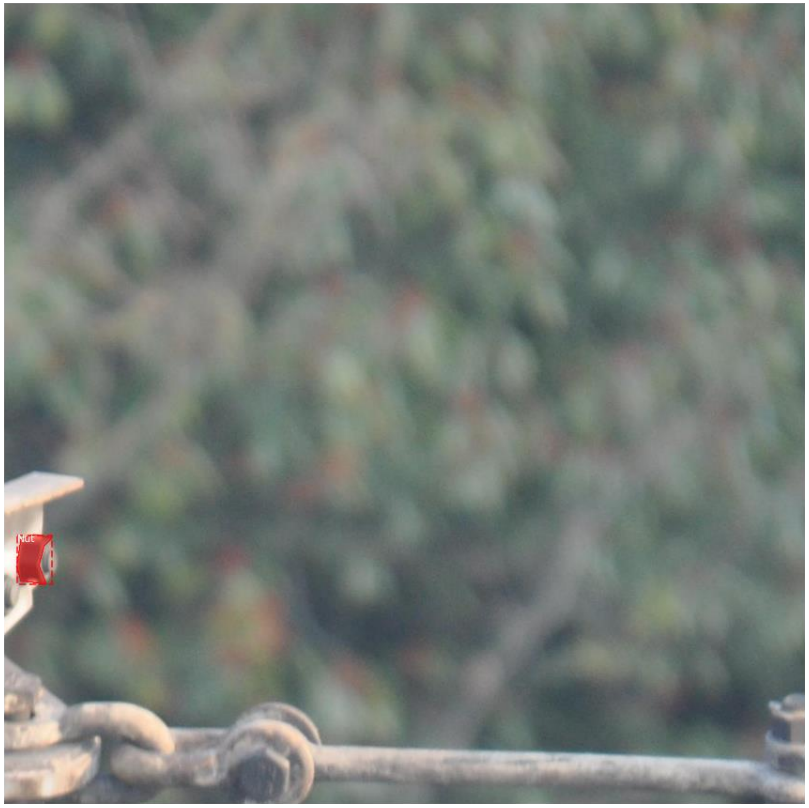
Predictions



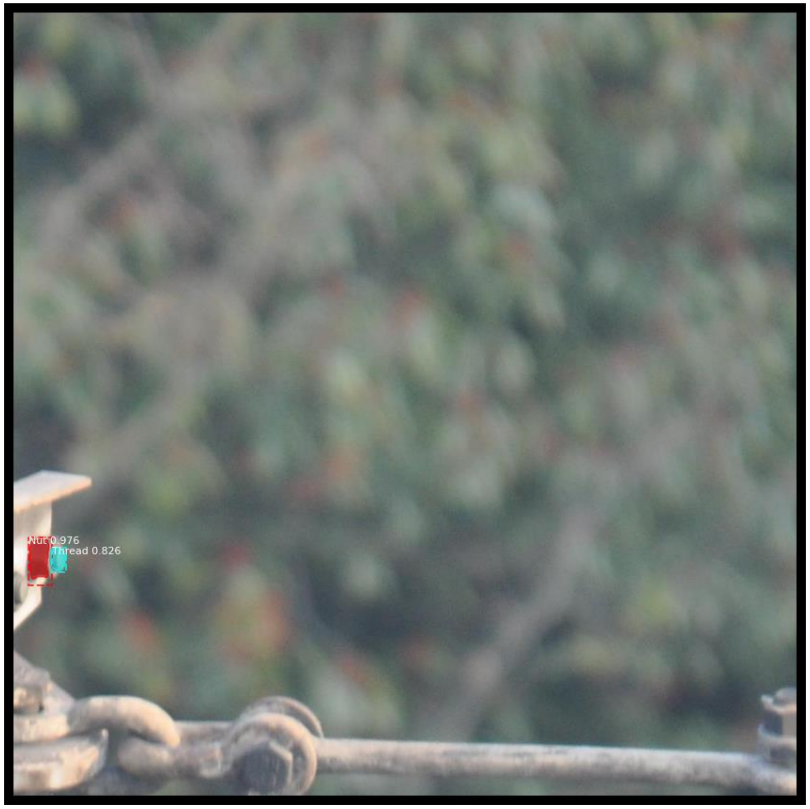


Predictions

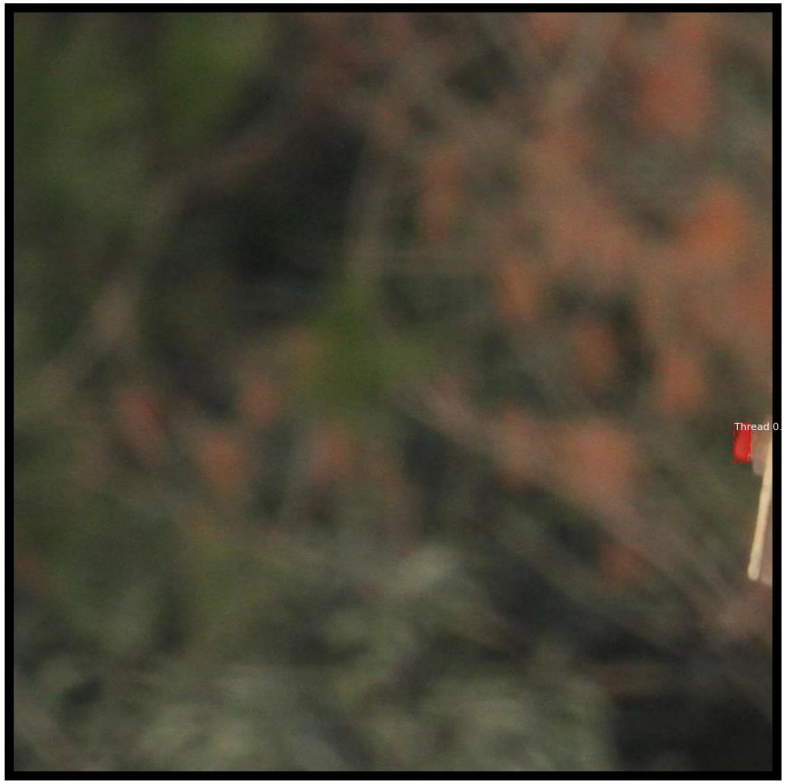


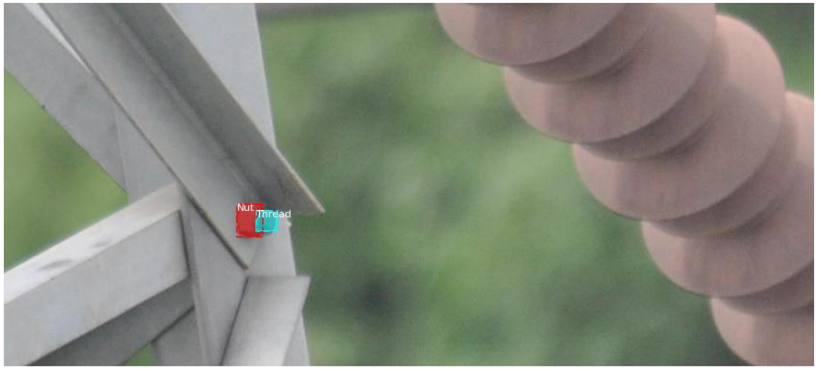


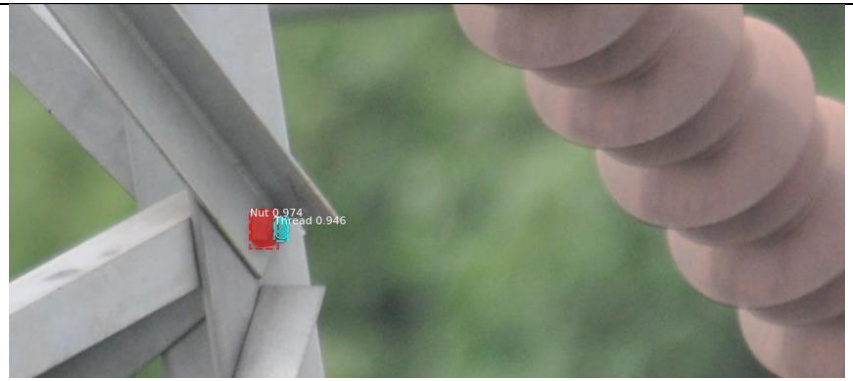
Predictions



Predictions




Nut
Thread


Nut 0.974
Thread 0.946


Head

5. Limitations

- We did not merge the pictures to the original pictures.
- We only use 5% of the data for testing purposes due to RAM limitations.

6. Conclusions

In this work, we utilize the state-of-the-art Mask R-CNN model and adopt a transfer training strategy to detect and classify small objects from the drone data set. The model with the best performance is achieved by head-only training (#epochs = 40, learning_rate = 0.001), and can successfully recapitulate annotations by human. Moreover, our model also identifies items that are missed in the ground-truth. Therefore, we demonstrate in this project our abilities to apply and fine-tune a model for object detection and our understandings of basic machine learning techniques.

References

- MaskRCNN/balloon.py from Github.
- Run_prediction_fromChopped.ipynub provided by Dr. Da Yan
- Drone data set provided by Dr. Da Yan.