Automatic Image Labeling with Click Supervision on Aerial Images

Krittaphat Pugdeethosapol[1], Morgan Bishop[2], Dennis Bowen[3], Qinru Qiu[1]
[1]Department of Engineering and Computer Science, Syracuse University, Syracuse, New York, USA
[2]Air Force Research Laboratory, Rome, New York, USA
[3]Technergetics, Utica, New York, USA
Email: [1]{kpugdeet, qiqiu}@syr.edu,[2]morgan.bishop.1@us.af.mil,[3]dennis.bowen@techngs.com

***Abstract*—Manually generating annotated bounding boxes for object detection is time consuming. Although human-annotation is the most accurate approach, machine learning models can provide additional assistance. In this paper, we propose a human in a loop automatic image labeling framework focusing on aerial images with less features for detection. The proposed model consists of two main parts, prediction model and adjustment model. The user first provides click location to prediction model to generate a bounding box of a specific object. The bounding box is then fine-tuned by the adjustment model for more accurate size and location. A feedback and retrain mechanism is implemented that allows the users to manually adjust the generated bounding box and provide feedback to incrementally train the adjustment network during runtime. This unique online learning feature enables user to generalize existing model to target classes not initially presented in the training set, and gradually improves the specificity of the model to those new targets online. We demonstrate promising results on Neovision 2 Heli dataset. Compared to the state-of-the-art method, our prediction model achieves a higher detection rate, and our adjustment model improves the IOU by up to 45%**

***Index Terms*—object labeling, object detection, click supervision, online training**

I. Introduction

Object detection has become one of the proliferating research fields in recent years. With the convolution based network structure and novel backpropagation technique [1], models can be trained to directly extract features from images or videos for object detection and classification. Many state-of-the-art models [2] [3] [4] [5] [6] [7] have demonstrated impressive results. Although they have different training techniques and network architectures, what is common is that they all require carefully labeled data for supervised training. Obtaining a set of high-quality labeled data is one of the major challenges for those who have to build and train their own model using supervised learning. Although there are many available public datasets, i.e. [8], [9], and [10], training an application specific model requires additional domain specific data.

Creating labeled data for object detection is time consuming. According to [11], annotators take about 35 seconds to draw and annotate a bounding box in the ILSVRC dataset. Do this repeatedly for every object in the training image is a tedious task for human annotator and impairs their productivity. On the other hand, human cognition is necessary for robust detection, especially in the cases where salient image feature is missing. Examples of such cases are given in Fig 1. Conventional CNN based object detection model such as YOLOv3 [12] generates bounding boxes only when it has a relatively high confidence that the region is not part of the background. Given the aerial image in Fig 1, the confidence level reduces. Human cognition is much more reliable in these cases.

In this paper, we present a human-in-loop automatic image labeling model that can be used to improve the annotator's productivity while maintaining the labeling accuracy. The proposed model mainly focuses on aerial images (top-down view) which has less salient features for detection and classification. It adopts a hybrid approach to automatically generate a bounding box around the object identified by a user click. The contribution of the paper is summarized as the following:

- We present a framework that fuse human object detection capability with the machine intelligence in feature extraction and semantic understanding to improve annotators productivity in labeling.
- We improve the feature pyramid in the YOLOv3 model to enhance the detection of small objects.
- An adjustment network is attached to the original detection model to dynamically learn how to improve the bounding boxes and adapt to new target classes that are not in the training set
- Experimental results show that we can improve IOU in prediction model by 35.6% in average. A further improvement of up to 45% can be reached after applying the adjustment network. The experimental results also show that we can utilize the feedback from users to incrementally train the model during runtime even with very small samples.

The rest of the paper is organized as the following. In section II, we review the existing methods that has been used to help training object class detectors. This is followed by Section III that provides details of our framework. Section IV describes our experimental steps and analyzes the results. Finally, Section V concludes this work.

II. Related Works

Human in a loop click annotation method has been used to help training object class detectors in many researches [13] [14] [15] [16] [17].

978-1-7281-6926-2/20/$31.00 ©2020 IEEE

Fig. 1. Example of aerial images.

Papadopoulos et al. [16] utilized crowdsourcing framework (Amazon Mechanical Turk) to collect click annotations. Annotators must pass a simple qualification test before they can proceed to the real annotation tasks which are divided into batches of 20 images. During the test, annotators will be asked to click as close as possible to the center of synthetic polygons. The purpose of the test is to ensure that annotators can locate the center of the object regardless of the shape. The click annotations are then used to incorporate into a reference Multiple Instance Learning framework designed for weakly supervised object detection [8] results in improvements of object class detectors. In 2011, Wah et al. [17] proposed a visual recognition system which combined a computer and a human into a single system to identify the class of the objects. The system asked the annotator to provide click location on the specific part of the image along with answering binary questions. The system will predict the most likely class based on the given information. The procedures will continue until the correct class was presented.

Other than object class detection, user click information has also been used in semantic segmentation [18] [19] [20] [21]. For example, Bearman et al. [21] combining user click information with state-of-the-art Convolutional Neural Network (CNN) for semantic segmentation by adding in training loss function to help infer the boundary of the objects.

Either object class detection or semantic segmentation requires the CNN to incorporate with user click location to complete the tasks. There are many state-of-the-art methods for class object detections. They can be divided into two main categories, one-stage methods [4] [7], and two-stage method [3]. In one-stage method, the model processes and predicts object bounding boxes regularly across the image, while in the two-stage method, the model will generate location proposals and then classify each proposal into one of the object classes or the background. There are trade-offs between these two methods in terms of speed and accuracy, the one-stage method is faster while the two-stage method is more accurate.

In this paper, we choose the one-stage method and incorporate it with user click location. More specifically we choose YOLOv3 [12] as our based model due to its speed and state-of-the-art accuracies.

III. Hybrid Human-Machine Labeling Framework

In this section, we introduce the overall architecture of the proposed framework. Its input is the image and the (x, y) coordinates of user click, which indicate the location of a target object. Its output is the bounding box of the target object. The framework can be divided into two main parts, prediction model and adjustment model. The prediction model generates the bounding box of the object selected by the user click, while the adjustment model refines the predicted bounding box to better fit the target object. The overall architecture is shown in Fig 2.

A. Prediction Model

The prediction model can be further divided into three main parts, a backbone network in which Darknet-53 is used, a Feature Pyramid Network [7], and a post-processing module, which is only used during the inference. During the inference, features extracted from different layers in the backbone network will be combined in the feature pyramid network to get the bounding box predictions. The Post-processing module is used to filter and select the final predicted box based on the user click information.

1) Backbone Network: Darknet-53 is used for backbone network for its high speed and state-of-the-art performance. It is a hybrid approach that combines Darknet-19 [4], residual network, and skip connection techniques. This results in 53 convolutional layers using only $1x1$ or $3x3$ kernels. The residual block replicates itself $1x, 2x, 4x$, and $8x$ times at each part of the network. The output size of the residual block is always the same as the input because of padding.

2) Feature Pyramid Network: We apply the feature pyramid network on top of Darknet-53. The last three feature maps of residual block are used as the inputs of feature pyramid network. Each feature map is up-sampled by $2x$ to match the size of the output of earlier layers. We use fusion to combine the up-sampled high level features with the low level features coming from the darknet. With this technique, the combined features allow us to extract information at higher resolution, and help to locate and detect the object with different scales and different feature representations. While YOLOv3 predicts boxes at three different scales and using concatenation when merging. Our unique feature pyramid network can identify small objects in the image more efficiently.

3) Post-processing: Post-processing module is used during inference. The goal of this module is to find the best bounding box prediction based on user click information. In traditional object detection methods, a threshold of the detection confidence must be set. If the threshold is too high, the model may not predict any boxes, and if the threshold is too low, the model will generate multiple boxes at the same location which sometimes makes it difficult to select the best one. The aerial images that we target at have low resolution and most of the bounding boxes will be filtered out due to a low confidence. However, a user click will significantly boost the detection confidence of the bounding boxes located at the clicked area. Instead of increasing the confidence of those bounding boxes, we lower the confidence threshold to keep more detected boxes, then use the post-processing algorithm to filter out the boxes not adjacent to the click location.

The input of the post-processing is all bounding boxes generated by the backbone and feature pyramid network, albeit

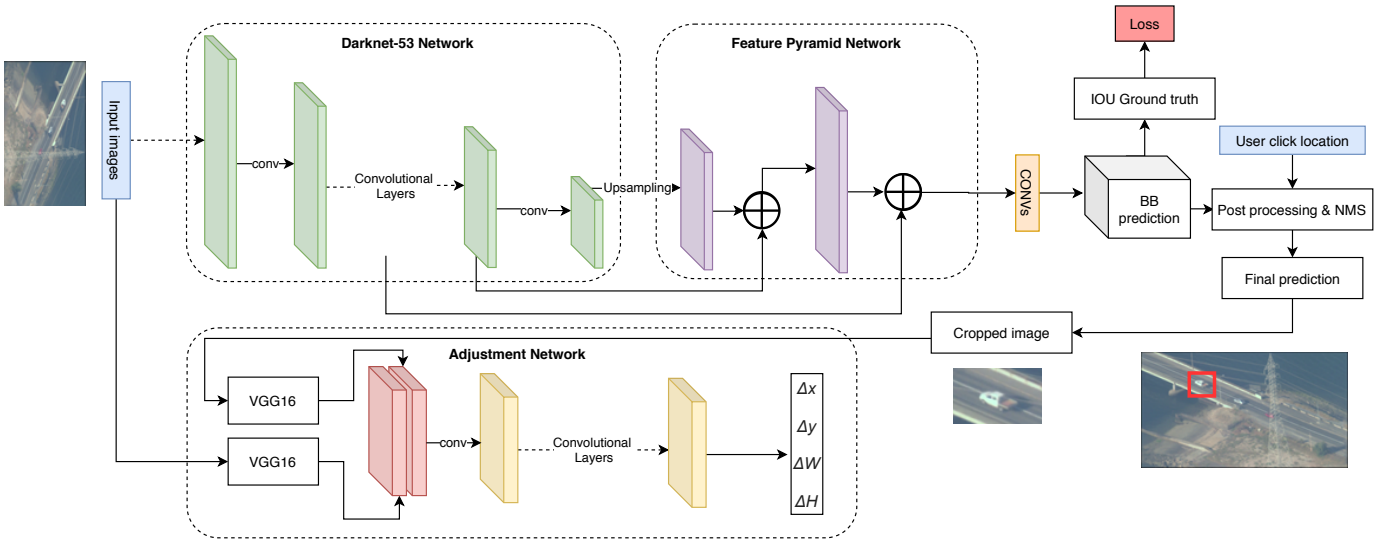


Fig. 2. The overall architecture of proposed work. There are two mains part, prediction model on the top section and adjustment model on the bottom There are two type of input to the model which include image and user click location. Image is used to feed in for both prediction and adjustment model, while user click location is used for post processing to find the final bounding box prediction. The prediction model predicts bounding based on user click location, while adjustment model predicts how much the predicted box should adjust to fit more to the target object.

with low confidence. Overlapped boxes have been filtered using Non-Maximum Suppression (NMS). With the assumption that there is exactly one object at the click location and the user already knew the class of the object; the post-processing module filters the input bounding boxes and selects the most fit one. The algorithm is shown below:

First, we filter boxes that do not contain the user click point. Then we filter boxes whose class information does not match the target class of the user click. In some cases, there will be no boxes left after the filtering. If this happens, we will alternatively select boxes belonging to other classes as an approximation since the model may miss classifying the objects that have similar features, e.g. bus as train, helicopter as an airplane, etc. The possible error led by such approximation can be mitigated by the adjustment network later. In the next step, we calculate the Euclidean distance between the center of boxes and user click location, sort and select the top 30% of remaining boxes that have the shortest distance. Finally, we select the box that has the highest detection score.

4) Training and Inference: Following YOLOv3, we divide an image into SxS grids, each grid predicts 9 anchor boxes with different aspect ratios and predifined width and height using k-mean clustering algorithm.

For each box, the model predicts 5 values: x, y, width, height, and confidence score. As a result, the final prediction features are encoded into a $SxSx45$ tensor, where $S = imagesize/16$ in our model.

We use a pre-trained Darknet-53 network on the MS-COCO dataset [10] as our backbone network. The prediction model is trained using three losses: location of boxes, width and height of boxes, and the confidence score of the box showing the probability of having an object. The loss equations are defined below:

$$L = \alpha * L_{xy} + \beta * L_{wh} + \gamma * L_{conf} \tag{1}$$

$$L_{xy} = \sum_{i=0}^{B^+} (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \tag{2}$$

$$L_{wh} = \sum_{i=0}^{B^+} (w_i - \hat{w}_i)^2 + (h_i - \hat{h}_i)^2 \tag{3}$$

$$L_{conf} = -\sum_{i=0}^{B^+} p_i log(p_i) + \delta * -\sum_{i=0}^{B^-} p_i log(p_i) \tag{4}$$

where α, β, γ and δ are weighted parameters for each part of the loss. We use a square loss for both location and size of the box. It only considers the set of boxes, denoted as B^+, which are responsible for the ground truth (i.e. IOU between the predicted box and ground truth more than a threshold). For confidence loss, we use cross entropy loss where p_i is an object probability calculated using $sigmoid$ activation function, It is calculated only for the set of boxes, denoted as B^-, which are not responsible for the ground truth.

Instead of predicting x, y, $width$, and $height$ directly, we predict x and y in relative to the location of the grid cell using $sigmoid$ activation function and predict width and height using the anchor boxes [4].

During inference, click location can be asked in two ways. [16] ask annotator to click on the center of the object while in [21] ask to click anywhere on a target object. In this paper, we choose to ask annotators to click anywhere on the target due to its feasibility in real applications. Some of the images may have small objects, which will be hard for annotators to click exactly at the center. Furthermore, concentrating on clicking at

Algorithm 1 Post-processing steps

Input

Bounding boxes generated from model: $B = B^i_{(x,y,w,h,c,s)}$ where $i = 0, 1, 2, ...n$

User click location and predefined class: $P = (p_x, p_y, p_c)$

Output

Final predicted box at user click location: $F = (f_x, f_y, f_w, f_h)$

List L: list of boxes that contain user click location

List M: list of boxes that contain user click location and match pre-define class

for $b \in B$ **do** select boxes that contain user click location
  $x_{min} b_x - b_w/2$, $x_{max} = b_x + b_w/2$
  $y_{min} = b_y - b_h/2$, $y_{max} = b_y + b_h/2$
  if $x_{min} \leq p_x \leq x_{max}$ and $y_{min} \leq p_y \leq y_{max}$ **then**
    $L.add(b)$
  end if
end for
for $l \in L$ **do** filter out boxes that don't match pre-define class
  if $l_c == p_c$ **then**
    $M.add(l)$
  end if
  if $M == \emptyset$ **then**
    $M = L$ (use boxes from other classes)
  end if
end for
for $m \in M$ **do** find boxes that have a minimum distance to a click location
  calculate Euclidean distance between center of a box and user click location
  $d_m = \sqrt{(m_x - {p_x}^2 + (m_y - p_y)^2}$
end for
sort(M by d_m): sort List M by distance from smallest-largest
$M = top_{30}(M)$: keep 30% smallest distance
$F = max(M)$: select the box that has the highest score
return F

the center will slow down the annotation process and reduce the human productivity.

B. Adjustment Model

The performance of the aforementioned model is highly determined by the training set. If the annotated image deviates from the training image, accurate bounding boxes may not be generated. Given the initial bounding box found by the aforementioned prediction model at the user click location, the goal of the adjustment model is to dynamically adapt to the testing data and adjust the size and shape of the box to be more precise over the target object. This is achieved by allowing the user to manually adjust the initially predicted bounding box to the correct size, and to collect the difference, ($\Delta\hat{x}$, $\Delta\hat{y}$, $\Delta\hat{W}$, and $\Delta\hat{H}$), between the predicted and corrected boxes. The difference will be referred to as the ground truth adjustment and will be used to train the adjustment model.

The adjustment model receives two inputs, the input image and the image of the target object cropped based on the predicted bounding box. Both images pass through VGG16 to extract features. The stacked features run through multiple convolution layers before finally used to predict the bounding box adjustment $\Delta\hat{x}$, $\Delta\hat{y}$, $\Delta\hat{W}$, and $\Delta\hat{H}$. The total loss to train the model is defined as follow:

$$L = L_{\Delta x} + L_{\Delta y} + L_{\Delta W} + L_{\Delta H} \tag{5}$$

where smoothed L1 loss is used for each loss:

$$L_{...} = \begin{cases} 0.5d^2 & |d| < 1 \\ |d| - 0.5 & otherwise \end{cases} \tag{6}$$

where d is the difference between the predicted adjustment (Δx, Δy, ΔW, and ΔH) and ground truth adjustment ($\Delta\hat{x}$, $\Delta\hat{y}$, $\Delta\hat{W}$, and $\Delta\hat{H}$). We maintain an adjustment network for each object class. Compared to Darknet or VGG net, the adjustment model is a relatively small and low-cost and hence is more suitable for online training.

The significance of the adjustment model is beyond refining the size and the location of the predicted bounding boxes. It also enables transfer learning. As we will show in the experimental results, the adjustment model only needs a few samples in order to adapt. By leveraging this property, we can apply the learned prediction model of one target class to predict the bounding boxes of objects of another target class that has not been included in the training set, as long as the two have similar features, and apply the adjustment model to adapt to the correct bounding boxes. For example, if we have a trained prediction model for trucks, but not for cars. We can still use the existing truck model to generate bounding boxes of cars at user click location. At first, the predicted box will not fit perfectly to the object because all observed features will be interpreted as trucks and be fit into a truck bounding box. Based on the user feedback, the adjustment model will gradually learn to adjust the bounding box specifically for the "cars".

The adjustment network is especially useful when the labeled data of a specific target class is limited. Training the adjustment model (rather than the prediction model) for this class will be more effective as there will be less overfitting. Furthermore, it will be more difficult to locate an object than adjusting a box that has already existed.

IV. Experiments

In this section, we describe the details of the experiments, including the dataset, the implementation details, and evaluation metrics. The experimental results will be presented for prediction models compared to a state-of-the-art method.

A. Experimental Setup

For prediction model, the pre-trained Darknet-53 on MS-COCO is used as the backbone network. We then fine-tuned the model on Neovision2 Heli dataset [22]. We chose this dataset to mainly focus on aerial images.

TABLE I
STATISTICS OF NEOVISION 2 HELI DATASET.

Class	Number of boxes	
	Train	Test
All	8839	17626
Car	3791	8065
Boat	251	1150
Helicopter	23	39
Person	1196	3867
Container	654	2824
Cyclist	240	426
Plane	2576	792
Tractor	12	130
Truck	96	243
Bus	0	90

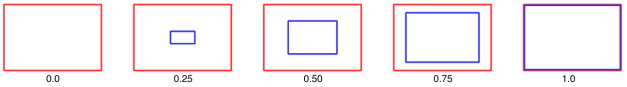


Fig. 3. Generated user clicks location area.

The dataset contains 32 video clips for training and 37 video clips for testing. The videos were filmed by a helicopter over the Los Angeles area and have 10 classes: car, boat, helicopter, person, container, cyclist, plane, tractor, truck, and bus. The annotation of videos is divided into frames. We only choose the frames and boxes that are not part of the Don't Care Regions (DCR), not ambiguous, and have confidence score equal to 1.0. With these criteria, there are no ground truth boxes left for "buses", so we exclude this class during the training. The total number of boxes for each class in training and testing sets are shown in table I.

We compare our model to a state-of-the-art YOLOv3 model [23]. Both models are trained on NVIDIA TitanX (Pascal) and Tensorflow r1.13. In the experiment, we set α, β, γ, and δ to 5.0, 5.0, 1.0, and 0.5 respectively.

During inference, we lower the model threshold at predict phase to 0.01 to get additional bounding boxes before applying post-processing to select the final prediction. The locations of user click were generated using uniform distribution with a specific range of the lowest and highest value. We used scale of 0, 0.25, 0.5, 0.75, and 1.0. At scale 0, we assumed that a user click is located at the center of the ground truth box. At scale 0.25, 0.5, and 0.75, user click locations were randomly selected within a rectangular area whose size is 25%, 50% and 75% of the ground truth bounding box located at its center. Obviously the larger the rectangular area is, the more possible the user click will deviate from the center. And at scale 1.0, user click locations can be anywhere within the ground truth box. Fig 3. shows the area where user click locations are generated at different range.

B. Evaluation Metric

To measure the accuracy of our predicted boxes, we use intersection of union (IOU) as the evaluation metric. If no predicted boxes were generated at user click locations, possibly due to very low detection confidence, the IOU is considered to be 0.

C. Prediction Model Results

We performed experiments on Neovision 2 Heli dataset, where we trained the model while considering all classes and each class individually. The threshold of 0.01 and IOU of 0.5 were used during the inference. We compared our model with a state-of-the-art YOLOv3 in which post-processing technique was applied for both approaches. The randomly generated user click location at scale of 0, 0.25, 0.5, 0.75, and 1.0 were used in the experiments. As mentioned above, "bus" class was excluded in this experiment since there is no labeled training data.

The results are shown in table II. Each row represents an average IOU of corresponding class objects at different user clicks. and average percentage improvement. Compared to the YOLOv3, our models have better IOU in every class and user click scale. This is because our model has a better detection rate with the help of user click information. We can also observe that the IOU dropped when the range of the user click expanded. This mean that the user click location plays an important role in the final prediction. The closer the user clicks locate near the object center, the better the predicted box can be.

Additionally, we tested the model on aerial video which is not part of the Neovision2 Heli dataset. Fig 4. shows the result when applying post-processing technique to get the final prediction box. From left to right, the first image represents all possible bounding boxes when the threshold has been lowered. The second image shows all boxes that contain user click location while the third image shows the top 30% of all boxes that have a minimum distance from click location. Finally, the last image shows the final prediction box which has the highest score. Fig 5. shows additional results at the final prediction.

D. Adjustment Model Performance

The second experiment evaluates the performance of the adjustment model. In this experiment, we assume the center click. We first apply the prediction model to obtain the initial bounding boxes. The results are then split into 70:30 ratio for training and testing. The adjustment model is trained while considering all classes and each class individually.

The results are shown in Table III. Each row represents the average IOU of corresponding class before and after applying the adjustment model. The IOU is calculated only if predicted boxes were generated at user click locations. It shows that, with adjustment model, IOU can be improved up to 45% in average.

We also evaluated the transfer learning capability of the adjustment network by using the pre-trained prediction model from different classes that have similar features to predict a new class of objects that has not been trained. We trained four prediction models using different training sets that consists of only: (1) cars, (2) all vehicles except the cars, (3) all classes in Table III except the cars, (4) all classes in Table III. Then

TABLE II
PREDICTION MODEL RESULTS.(A) YOLOV3 (B) OUR MODEL

Click location	0			0.25			0.50		
	(a)	(b)	% improvement	(a)	(b)	% improvement	(a)	(b)	% improvement
Class									
All	0.2446	0.3647	49.10	0.2399	0.3615	50.69	0.2028	0.3319	63.66
Car	0.2519	0.3095	22.87	0.2506	0.3071	22.55	0.2203	0.2718	23.38
Boat	0.1298	0.1349	3.93	0.1262	0.1614	27.89	0.1077	0.1208	12.16
Helicopter	0.2442	0.2790	14.25	0.2442	0.2790	14.25	0.2301	0.2790	21.25
Person	0.0804	0.1879	133.71	0.0728	0.1763	142.17	0.0653	0.1297	98.62
Container	0.1359	0.2579	89.77	0.1292	0.2500	93.50	0.1086	0.2167	99.54
Cyclist	0.4216	0.4884	15.84	0.4225	0.4764	12.76	0.3580	0.4638	29.55
Plane	0.3844	0.4172	8.53	0.3785	0.4141	9.41	0.3128	0.3447	10.20
Tractor	0.1720	0.1828	6.28	0.1576	0.1798	14.09	0.1405	0.1537	9.40
Truck	0.1785	0.2965	66.11	0.1708	0.2965	73.59	0.1468	0.2508	70.84

Click location	0.75			1.00			Average		
	(a)	(b)	% improvement	(a)	(b)	% improvement	(a)	(b)	% improvement
Class									
All	0.1320	0.2449	85.53	0.0846	0.1717	10.96	0.1808	0.2949	63.15
Car	0.1491	0.1806	21.13	0.0968	0.1193	23.24	0.1937	0.2377	22.67
Boat	0.0695	0.0861	23.88	0.0410	0.0628	53.17	0.0948	0.1132	19.36
Helicopter	0.1921	0.2194	14.21	0.1113	0.1830	64.42	0.2044	0.2497	21.28
Person	0.0471	0.0770	63.48	0.0334	0.0482	44.31	0.0468	0.1238	164.57
Container	0.0761	0.1439	89.09	0.0538	0.0899	67.10	0.1007	0.1917	90.31
Cyclist	0.2302	0.3295	43.14	0.1580	0.2031	28.54	0.3181	0.3922	23.32
Plane	0.2018	0.2279	12.93	0.1221	0.1559	27.68	0.2799	0.3120	11.45
Tractor	0.1246	0.1247	0.08	0.1081	0.1087	0.56	0.1406	0.1499	6.67
Truck	0.1003	0.1876	87.04	0.0750	0.1302	63.60	0.1343	0.2323	73.01
Average							0.1694	0.2297	35.60

Fig. 4. An example of post-processing output at each step.

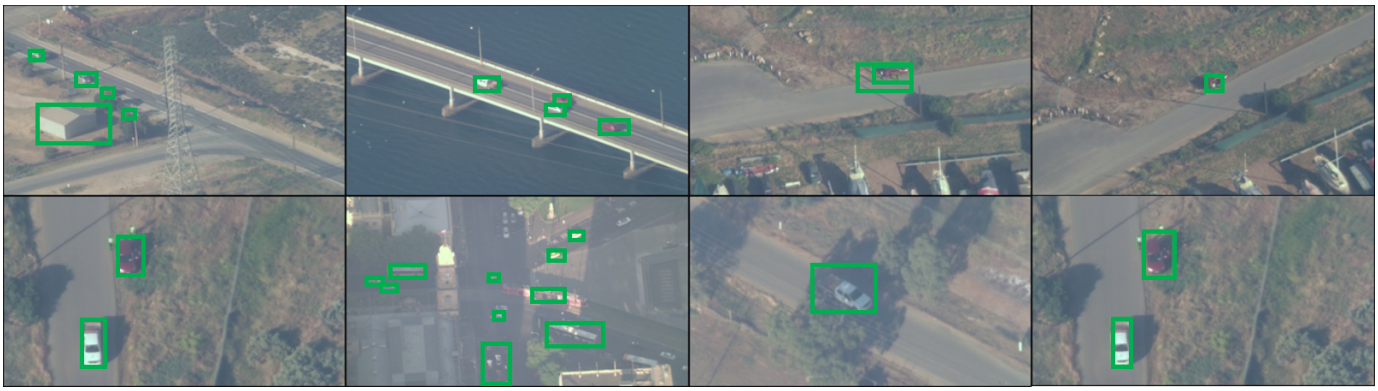
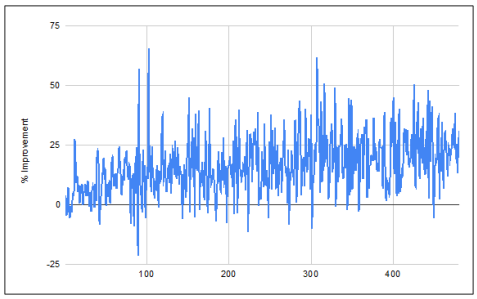
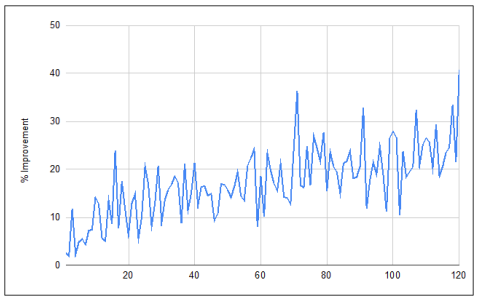


Fig. 5. Examples when applying our prediction model and post-processing technique.

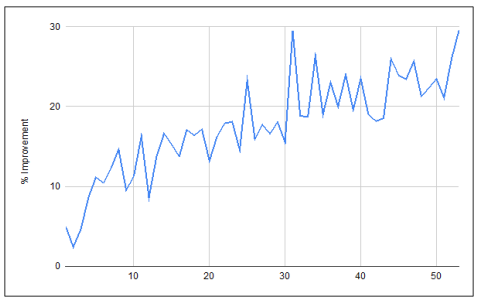
Fig. 6. Examples when applying adjustment model. Left: before adjustment. Right: after adjustment. Red: ground truth box. Blue: predicted box.



(a) N=10



(b) N=40



(c) N=90

Fig. 7. Incremental learning results. y-axis represent percent improvement, x-axis represent number of batch where the model has been retrained when new data is available. (a) size 10, (b) size 40, (c) size 90. The size indicates how many new feedback boxes will trigger the model to retrain itself.

we apply the prediction model to predict the bounding box of cars followed by the adjustment model that learns to fine tune the bounding boxes. Table IV. shows the prediction results of the four configurations. Each row represents the detection rate of the prediction model, the IOU before adjustment, and the IOU after adjustment.

As we can observe, when the model is trained with data belong to all other classes except the car (i.e. 3^{rd} row in Table IV), we got the lower detection rate and IOU before and after the adjustment. This happens because there were classes in the training data that is totally different from cars. Using a prediction model learned from those examples results in lower overall performance.

We can also see that we got promising results if the model is trained based on data from all other vehicle classes except the car (i.e. scenario 2 in Table IV). Even though the IOU without adjustment is lower than that of scenario 1 (i.e. prediction model trained using car images), it has a higher detection rate and the IOU can be improved by adjustment. This result confirms that we can adjust the model trained for other classes to predict the bounding box of new target class, if they share some similarities. Fig 6. shows some example of bounding boxes before and after apply adjustment model. The same results can also be observed for other classes.

Since the adjustment model is a relatively low-cost network, it can utilize the feedback from users to incrementally train the model during runtime. Fig 7. shows how adjustment model improves the IOU over the incremental learning procedure. In this experiment, the target class is the "car" class and the prediction model was trained using the image of cars. We use the ground truth bounding box of the testing samples as the

TABLE III
ADJUSTMENT MODEL RESULTS.

Training data	Average IOU		Percentage Improvement
	Before	After	
All	0.4416	0.5650	27.94
Car	0.4975	0.6095	22.51
Boat	0.4240	0.7199	69.78
Helicopter	0.3858	0.7419	92.30
Person	0.4649	0.6395	37.55
Container	0.4652	0.6410	37.79
Cyclist	0.5094	0.6382	25.28
Plane	0.4997	0.7252	45.12
Tractor	0.2246	0.3397	51.24
Truck	0.3665	0.5205	42.01
Average	0.42792	0.61404	45.15

TABLE IV
ADJUSTMENT MODEL RESULTS ACROSS CLASSES.

Training Data	Average IOU		Detection rate
	Before	After	
(1) Car	0.4975	0.6095	60.01%
(2) Vehicle except car	0.3337	0.6098	82.00%
(3) All except car	0.3432	0.4580	26.00%
(4) All	0.4290	0.5593	84.80%

user feedback. The adjustment network is incrementally retrained every N samples, and N varies from 10 to 40 and 90. The plots show the improvement in IOU after applying the adjustment model. As we can see even with a small batch N=10, the adjustment network adapt to the testing samples after 20 batches and start to deliver around 10% improvements over the original prediction model. For all different N, after about 3,000 samples, the adjust network can about 25% of improvements in average

V. Conclusion

In this paper, we proposed a human in a loop automatic image labeling model. The combined prediction, post-processing, and adjustment model can be used to streamline the analyst's job in annotating images or videos. The model focuses on aerial images, which has less salient features for detection. We demonstrate promising results on Neovision 2 Heli dataset with a comparison to YOLOv3.

Also, by leveraging user click information and the adjustment model, we can improve the overall IOU and extend the framework during runtime to adapt to new classes whose labeled training data is not readily available.

VI. Acknowledgment and Disclaimer

This work was received and approved for public release by AFRL on 04/30/2020, case number 88ABW-2020-1588. Any Opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL or its contractors.

References

[1] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural computation*, vol. 1, no. 4, pp. 541–551, 1989.

[2] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[5] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[6] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[7] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[8] R. G. Cinbis, J. Verbeek, and C. Schmid, "Weakly supervised object localization with multi-fold multiple instance learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 1, pp. 189–203, 2016.

[9] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[10] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[11] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015.

[12] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[13] E. Teng, R. Huang, and B. Iannucci, "Clickbait-v2: Training an object detector in real-time," *arXiv preprint arXiv:1803.10358*, 2018.

[14] J. Lee, S. Walsh, A. Harakeh, and S. L. Waslander, "Leveraging pre-trained 3d object detection models for fast ground truth generation," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2504–2510.

[15] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari, "We don't need no bounding-boxes: Training object class detectors using only human verification," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 854–863.

[16] ——, "Training object class detectors with click supervision," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 6374–6383.

[17] C. Wah, S. Branson, P. Perona, and S. Belongie, "Multiclass recognition and part localization with humans in the loop," in *2011 International Conference on Computer Vision*. IEEE, 2011, pp. 2524–2531.

[18] S. Bell, P. Upchurch, N. Snavely, and K. Bala, "Material recognition in the wild with the materials in context database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3479–3487.

[19] T. Wang, B. Han, and J. Collomosse, "Touchcut: Fast image and video segmentation using single-touch interaction," *Computer Vision and Image Understanding*, vol. 120, pp. 14–30, 2014.

[20] J. Liew, Y. Wei, W. Xiong, S.-H. Ong, and J. Feng, "Regional interactive image segmentation networks," in *2017 IEEE international conference on computer vision (ICCV)*. IEEE, 2017, pp. 2746–2754.

[21] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei, "What's the point: Semantic segmentation with point supervision," in *European conference on computer vision*. Springer, 2016, pp. 549–565.

[22] L. Itti, "Neovision2 annotated video datasets," 2014.

[23] Y. Yang, "tensorflow-yolov3," 2019. [Online]. Available: https://github.com/YunYang1994/tensorflow-yolo