Weakly Supervised Segmentation of Small Buildings with Point Labels

Jae-Hun Lee ChanYoung Kim Sanghoon Sull
School of Electrical Engineering, Korea University
{jhlee|cykim}@mpeg.korea.ac.kr sull@korea.ac.kr

Abstract

Most supervised image segmentation methods require delicate and time-consuming pixel-level labeling of building or objects, especially for small objects. In this paper, we present a weakly supervised segmentation network for aerial/satellite images, separately considering small and large objects. First, we propose a simple point labeling method for small objects, while large objects are fully labeled. Then, we present a segmentation network trained with a small object mask to separate small and large objects in the loss function. During training, we employ a memory bank to cope with the limited number of point labels. Experiments results with three public datasets demonstrate the feasibility of our approach.

1. Introduction

Recently, a variety of image segmentation methods based on deep learning have shown remarkable performance. However, most of them require vast amounts of pixel-level labels, which are quite expensive [3, 6, 29]. Further, labeling of small objects requires more attention than that of larger objects. As in MS COCO [17], some evaluation criteria of object segmentation consider the size of objects(small/medium/large). In this paper, we focus on the segmentation of objects or buildings, especially for small objects given their weak labels that do not require precise labeling.

One of the popular methods for labeling an object is based on manual contour following, or drawing of a tight polygon by placing multiple points around the object [12]. The pixels enclosed by the contour or polygon are labeled as *objects*. This manual labeling seems doable for a large object. But, for a small object, it results in larger portion of mislabeled pixels than that of a large object. The precise labeling of a small object requires time-consuming delicate effort.

Detection result of small objects is not usually satisfactory compared with that of large ones, probably due to insufficient labels for small objects. However, detection of small buildings could be useful to some specific tasks. For example, considering a task of locating the position of an input satellite/aerial image when its position information is not available, the spatial relationship between detected small and large buildings can give useful clues to its position. In this paper, we propose a new easier point labeling scheme for small objects as well as a segmentation network which can be trained by using the point labels for small objects and traditional contour labels for large objects.

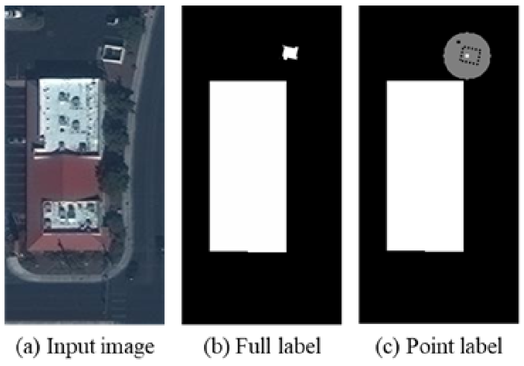

Figure 1. Point labeling of a small object. (a) Input image. (b) Full labels for both a small and a large object. (c) Point labels for the small object and a full label for the large object. In the gray circle which indicates unknown region, a small object point is shown in white, and its corresponding background point is shown in black.

Instead of trying to accurately label the contour of a small object, we label it using two points inside and outside the small object, respectively, as shown in Figure 1. Although the point label is less informative than the full label, the point labeling work requires less delicacy than the full label, reducing the annotation time as mentioned in [2]. Instead of clicking precisely on the corner points of the object, you can roughly click on arbitrary points inside and outside the object. In this paper, an object is called as small object if its area is smaller than the small object threshold T_s. Otherwise, it is treated as a large object. It is noted that a regular contour-based labeling is used for large objects. It is motivated by the map data in which a building is simply

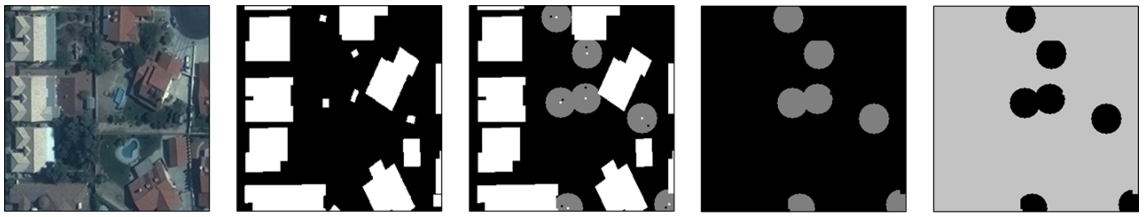

Figure 2. Point labeling of small objects. (a) Input image. (b) Full label for all objects. Object labels are shown in white, and background is shown in black. (c) Point labels in white for small objects with the corresponding black background points and gray unknown region. (d) Small object masked region M_s in gray contains unknown region, small object points and corresponding background points. (e) The region other than M_s is denoted as M_r in light grey.

labeled as a point without full labels, especially for small, school, or church buildings, and the point label is intended to indicate its location rather than its exact shape.

Contrary to existing segmentation methods employing a single same type of labels such as contours and scribbles, we use both point labels for small objects and full labels for large objects. Thus, it is important to properly process the different types of labels. As shown in Figure 1, large objects with full labels and small objects with white point labels are labeled as foreground. Labeling the remainder area as *background* results in mislabeling of small objects except their labeled small object points, whereas labeling it as *unknown* results in lack of information about background. Considering these, we propose a small object mask as illustrated in Figure 2.

In the case of pixel-level loss of semantic segmentation rather than object-level loss of instance segmentation, if we don't consider the size of an object during training, it can cause a bias in favor of large objects because the total area for small objects is usually much less than that of large objects. In other words, considering the point-wise loss and its average over whole points, small objects occupy small portion in the loss function compared to large object. Several methods attempted to give weights to small objects according to class [5, 14]. In particular, Jakub *et al.* [4] consider the size weight of an object during building detection of an aerial image. But, for our small-point label and large-full label dataset, it did not work well because the simple addition of weights caused circular artifact around the small object prediction. To extract information from small objects as much as possible and incorporate it into a network during training, there is a need for enabling the network to accumulatively remember the features of small object points from previous training iterations.

In this paper, we propose a weakly supervised segmentation network, assuming that small and large object labels are given as points and contours, respectively. Our proposed network employs the sampling of uncertain points and false positive small object points as well as a memory bank. We adopt a method of sampling points and classifying them, which can be applied to both point and full labels. We employ separate losses of small and large objects using a small object mask shown in Figure 2. In the small object masked region M_s containing only point labels, a memory bank is used to compensate for the insufficient information provided by the limited number of point labels. Additionally, to learn the contours of small objects, the image gradient is matched to the prediction gradient during training. In the remainder region M_r, the points are sampled by focusing on false positive small object points and uncertain prediction points. Our approach increases small object detection performance at the expense of slight decrease in large object detection performance.

Our contribution is as follows. First, we propose an easier labeling method for small objects. Second, to cope with the problem that model ignore small objects during training, we sample uncertain points and false positive small object points and consider small and large objects separately in the loss function. Third, to cope with the limited number of point labels, our network remembers and updates feature vectors of small object points based on a memory bank during training.

2. Related work

Weakly supervised segmentation means the methods to train segmentation networks by using the labels such as scribble, point and image-tag. Mai *et al.* [18] learn their segmentation network using image-tag labels. One of the point labeling methods [2] uses objectness prior [1] to alleviate local minima, where only the point portion of the target object is predicted as foreground class. In instance segmentation, point labels are used for object localization [16]. Wang *et al.* [28] adopt a point labeling for remote sensing imagery. ScribbleNet [30] is trained by scribble labels. To learn the contour of an object, it reflects the edge information in an image when computing the gradient of its prediction. These method uses a label-based loss only on point labels. For the rest of the regions, it utilizes pseudo-labels or loss function

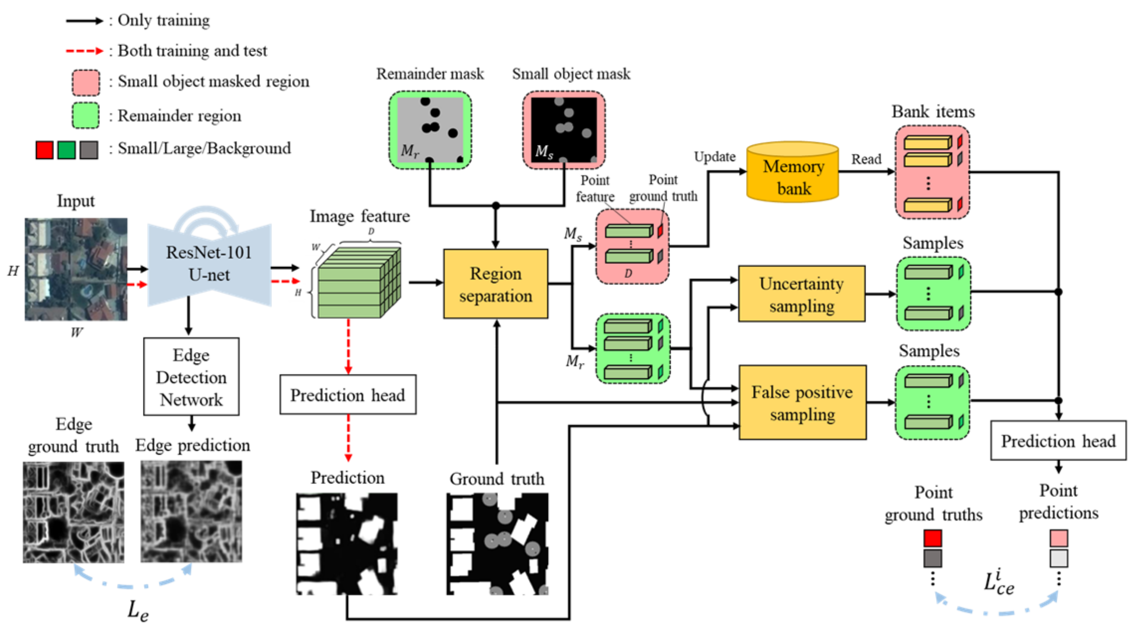

Figure 3. Overall architecture. In the small object masked region M_s, feature vectors are trained by utilizing a memory bank. Feature vectors and labels of the small object points and corresponding background points are feed to the memory bank, and updated with similarity score. In the remainder region M_r, we sample uncertain points using the entropy of class prediction probability. And, we sample false positive small objects using false positive score, which is defined by binary cross entropy of small object *vs* all other classes. The D-dimension point feature vectors from the point samples and the memory bank are input to the prediction head, which consists of three point-wise convolutional layers and outputs C-class prediction.

without labels. In this paper, labeled data is refined using a memory and point sampling.

Point feature refers D-dimensional feature vectors calculated from point sets, or sampled from the image feature map with $H \times W \times D$. Qui *et al.* [22] uses it for the semantic segmentation of point cloud data. Among the image segmentation methods, PointRend [15] samples point features as input to the multi-layer perceptron.

Memory bank is used for storing and reading features without updating stored data. For image synthesis, the style of a scene and object can be synthesized differently using a style code bank [26]. Class-specific real patches can be stored to compare real and generated images patch [27]. The memory bank can also be used for video segmentation [23], by storing feature maps and segmentation labels to train a sub-network. MoCo [10] uses a queue-based memory bank for self-supervised learning, and current data and old data as input to different encoders, having feature of stored data different from feature of current data. In contrast, we update stored feature similar to current feature.

Memory network usually reads and writes its item according to the similarity score [21, 25, 32]. They use the weighted sum of input and similarity score, and apply nonlinear functions afterwards. But some of the memory networks are similar to long short-term memory (LSTM) [7, 31] with trainable weights. Gong *et al.* [8] read a memory item according to the similarity score, but the memory itself is updated by minimizing the entropy of the score.

3. Our approach

In this section, we present our approach on the segmentation of objects, especially for small objects. As stated before, we assume that we are given point labels for small objects and full labels for large objects. First, we describe the small object mask which is used to separate small and large objects in the loss function (Section 3.1). And we describe how to strengthen the training of small objects by using point sampling (Section 3.2). We build a memory bank of feature vectors by sampling of labeled small object points (Section 3.3), and sample false positive small object points using false positive scores (Section 3.4). The total loss is shown in Section 3.5. Figure 3 shows the overall architecture of the proposed network.

3.1. Small object mask

For segmentation of an image, we use a loss for a whole image as follows:

$$L = \frac{\sum_i L^i}{N}, \tag{1}$$

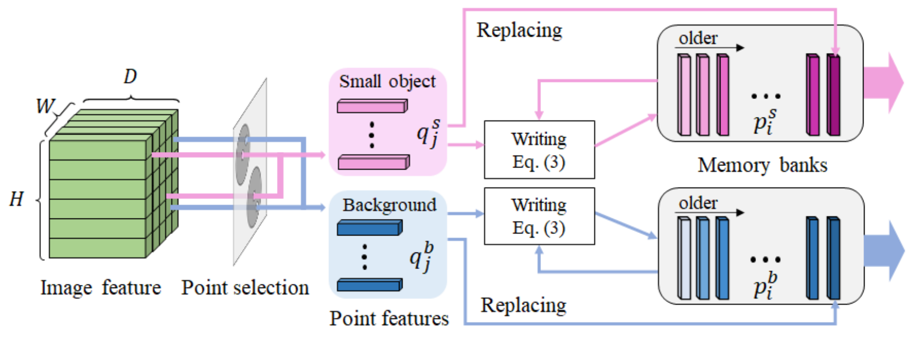

Figure 4. Memory bank update. We select all small object points and corresponding background points from small object masked region M_s. The update process consists of writing and replacing, which are performed separately for objects and background. For writing, all input point features with ground truth labels are queried to memory bank, and the memory bank items are updated by similarity scores. For replacing, N_K-oldest items are replaced with the random input points features. All items stored in memory bank are used for loss calculation.

where N is the total number of pixel points, and L^i is the loss from the i-th point. Alternatively, a set of sampled points can be considered as follows:

$$L = \frac{\sum_{i \in R} L^i}{N_R}, \tag{2}$$

where R and N_R denote a set of all points in the sampled region and its cardinality, respectively. Note that we can use multiple R, which makes it possible to consider each region differently in the loss function.

$$L = \frac{\sum_{i \in R_1} L^i}{N_{R_1}} + \frac{\sum_{i \in R_2} L^i}{N_{R_2}} + \ldots + \frac{\sum_{i \in R_r} L^i}{N_{R_r}}. \tag{3}$$

As stated in introduction, to appropriately balance the background and unknown, we separate the regions of small and large objects. Thus, for each small object, we consider a circled region centered at a point inside the small object. The union of all circles forms a small object mask M_s. Also, since we have a full label for the large object, the pixels corresponding to the large object is excluded from M_s. A remainder region M_r is defined as the whole image pixels excluding M_s. M_s or M_r correspond to R in equation (3). The small object mask M_s and the remainder region M_r are shown in Figure 2.

3.2. Point sampling

If the class probability entropy of the i-th point is high, this point can be seen as a uncertain point. PointRend [15] selects uncertain point samples during training, instead of using whole points. But it can omit points corresponding to small objects because it samples points sparsely, which affect detection performance on small objects. To overcome this weakness, we sample points of labeled small objects in M_s, and points predicted as small objects in M_r. The typical portions of uncertain points, points predicted as small objects, and points of labeled small objects are 50%, 25% and 25%, respectively.

By considering all point labels in M_s, all true small object points and their corresponding background points can be selected. However, the use of all points is not enough because the number of point labels in small objects is insufficient. Existing methods use larger weights to solve this problem [4, 5, 14]. In other words, they sample the same point repeatedly. Thus, it would be better to remember and correlate feature vectors of the limited number of true small object points we have seen during training. It is described in Section 3.3.

Contrary to small object points in M_s, the large number of points with labeled large objects is available in M_r. In this case, we sample only a portion, which includes uncertain points. Since all points in M_r do not have ground truth label of small object class, we select or sample false positive small object points. However, if we use two classes of *object* and *background* rather three classes of *small object*, *large object* and *background* used in this paper, the process of sampling can be time-consuming because of the difficulty of sampling false positive points. It is described in Section 3.4.

3.3. Labeled small object point samples

The problem due to the small number of point labels for small objects is alleviated by using a memory bank shown in Figure 4. We would like to fully utilize data we have seen during training, and we construct a module for storing features of previous inputs. For this purpose, we utilize the queue-based memory bank proposed in MoCo [10]. We modified the memory bank in the form of a $N_B \times D$ matrix, storing the N_B number of D dimensional feature vectors. In MoCo, the memory bank is used for negative data sampling in terms of contrastive learning, with its slowly up-

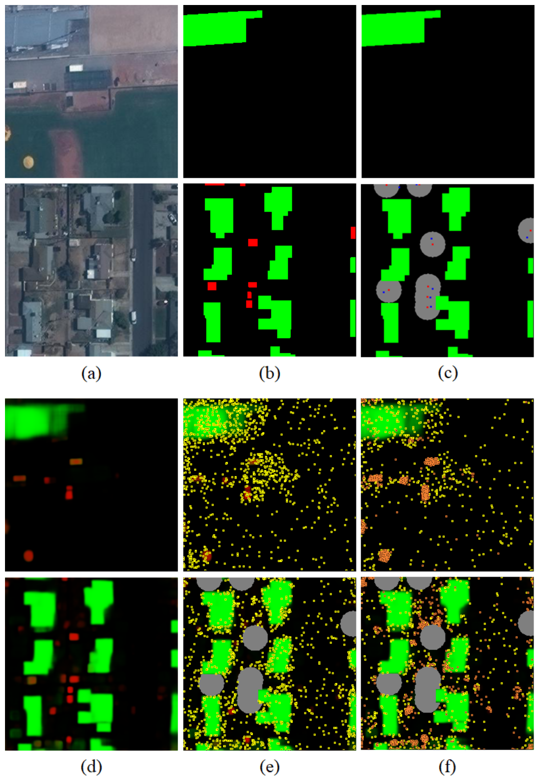

Figure 5. False positive small object samples with/without labeled small objects. (a) Input image. (b) Ground truth full labels for all objects. Large objects are shown in green, small objects are shown in red. (c) Ground truth point labels for small objects and full labels for large objects. Unknown region is shown in gray and the background points are shown in blue. (d) Prediction result during training. (e) 1024 yellow uncertain point samples obtained by using PointRend [15]. (f) 512 yellow uncertain point samples and 512 orange false positive small object point samples.

dated sub-network. However, because we use the memory bank for the diversity of labeled small object point samples, we need to consider the update of memory data features that are consistent with current mini-batch features from the latest-updated network.

We update the memory bank through two steps of writing and replacing. The writing step is building items in the bank using the weighted sum of the input query data, and the replacing step is replacing the oldest N_k items with random samples from the input query data. The queries for the memory bank consist of the features of small object points and corresponding background points. The number of query features N_Q are varied according to number of small objects in the image.

The query features q_j^c (j=1,...,N_Q) are written into the memory items p_i^c (i=1,...,N_B) using the equation:

$$p_i^c \leftarrow g_i p_i^c + (1 - g_i) \sum_j v_{i,j} q_j^c \tag{4}$$

where g_i is a parameter to control the gating value of writing. The similarity scores $v_{i,j}^c$ between the query features and memory items are calculated as follows:

$$v_{i,j} = \operatorname*{softmax}_j (p_i^c \cdot q_j^c) \tag{5}$$

which are summed to 1 over the query dimension j. Note that the query features and memory items are L_2 normalized.

The read policy of memory bank is different from those used in memory networks [7, 8, 21, 25, 31, 32]. While the memory network selectively reads memory based on the similarity or trained weight, we take all N_B items during training in order to prevent the memory bank from retrieving only similar features to the query features. Because the purpose of memory bank is retaining various features as much as possible, all items including dissimilar items should be read.

There are also various strategies for determining g_i. For the memory network with trainable weights [7, 31], they also learn the gating value g_i. But for the memory network using the similarity scores as weights [21, 25, 32], g_i is set to constant. Our memory bank can be seen as the similarity-based memory network in terms of write policy. But it would be better to use the gating values, g_i, which are determined by the queries and bank items.

Suppose there are multiple clusters for query features. If a bank item is far from all query features, the similarity scores are similar. The bank item is updated with the average value of all the queries with different clusters, which can be a non-interesting feature. It means that it is better not to trust the weighted sum $\sum_j v_{i,j} q_j^c$. In this case, g_i should be high. On the other hand, if an item is close to a certain query cluster, the similarity score for the query will be high. Then the weighted sum of queries can represent the query cluster, and it is good to trust the weighted sum, making the value of g_i low.

So, we set the entropy of similarity score to g_i. If the distribution of the scores is uniform, the entropy will be high. Conversely, if only certain score values are high, the entropy will be low. It implies that the entropy satisfies the properties needed for g_i. As a result, g_i is set to the entropy of similarity score:

$$g_i = -\frac{\sum_j v_{i,j} \log v_{i,j}}{\log N_B}, \tag{6}$$

where the entropy of similarity score is normalized to 1 using the entropy of the uniform distribution.

We maintain two different memory banks for small object points and their corresponding background points, and process them separately. For small object points, we denote the query features as q_j^s and memory items as p_i^s. For background points, we denote the query features as q_j^b and memory items as p_i^b. Note that the memory bank is used only in training process because it only affects the loss calculation.

3.4. False positive small object point samples

As described earlier, if we use two classes of *objects* and *background*, sampling of false positive small objects becomes time-consuming because we need to identify small objects from the class probability map. In other words, we need to binarize the probability map into object class and background class, check the area of each predicted instance, compare it with its ground truth, select prediction-ground truth pairs with maximum intersection-over-union (IoU). Finally, we select the predicted instance without matched ground truth instance. Note that this instance is false positive small object.

But we found that the false positive small object points can be obtained easily by predicting three classes of *small object*, *large object* and *background*. When we find false positive small object points, *large object* class and *background* class are combined as *non-small object* class. We calculate the false positive score with its ground truth value using binary cross entropy of *small object* class and *non-small object* class, and then N_S number of points with high false positive scores are selected.

As proposed in [9], detection of small and large objects can be trained in separate branches and combined later. They define different classes for small and large objects in ground-truth level, but not in prediction level. The sum of the probability of the small objects, large objects and background can be larger than 1. It requires the time-consuming prediction fusion process, which lead to time-consuming point sampling process. So instead of following the multi branch method in [9], we decided to use a single branch multi class approach for small objects and large objects, as shown in Figure 5. Large objects are shown in green whereas small objects are presented in red. Yellow uncertain points and orange false positive small object points are also shown in Figure 5. While the points in the uncertain samples [15] do not capture false positive small objects, our method captures false positive small object points well.

3.5. Loss function

Considering the small object masked region M_s and the remainder region M_r, the total loss function can be expressed as follows:

$$L = L_s + L_r. \tag{7}$$

The loss L_s for M_s is:

$$L_s = \frac{\sum_{i \in I_B} L_{ce}^i}{N_B} + \frac{\sum_{i \in M_s} (L_e^i + L_b^i)}{\text{Area}(M_s)}, \tag{8}$$

where I_B is a set of indices of the memory bank, L_{ce} is the cross entropy for the point samples from the memory bank. To precisely predict the regions or contours of small objects, we use the edge detection loss L_e and the smoothness loss L_b, as in ScribbleNet [30].

The loss L_r for M_r is:

$$L_r = \frac{\sum_{i \in R_S} L_{ce}^i}{N_S} + \frac{\sum_{i \in R_U} L_{ce}^i}{N_U}, \tag{9}$$

where we use the cross entropy of point samples obtained using the prediction uncertainty and the false positive score. The sampling locations and the number of the uncertain points are denoted as R_U and N_U, respectively, and the sample locations and the number of false positive small object points are denoted as R_S and N_S, respectively.

Segmentation can be seen as point-wise object detection. As suggested by PointRend [15], the uncertain point sampling can replace the use of whole points in training. If it is used without separation of small objects and large objects, it can miss the point predicted as a small object, thus causing false positive small object to be also missed, resulting in over-detection of small objects. Our proposed false positive small object point sampling and proposed memory bank for labeled small object point sample can solve this weakness of PointRend. The uncertain point sampling is the only supervision for labeled large object, which is the same as PointRend.

4. Experimental results

In this section, we first explain three public datasets used in our experiments: CrowdAI Mapping Challenge dataset [20], WHU building dataset [13] and Massachusetts buildings dataset [19]. Second, we describe a point labeling process for a given dataset. Finally, we present our experimental results. Implementation details and ablation studies (about point sampling rules, memory bank, network architecture, small object prediction method and the various values of N_K and r) are given in the supplementary material.

4.1. Dataset

We experiment with CrowdAI Mapping Challenge dataset [20] providing satellite images and labels for several small and large objects or buildings. The size of original images is 300×300, and the images are resized to 256×256. The number of training images is 280,741, and the number of validation images is 60,317. All objects are fully labeled.

The threshold for small objects T_s is set to 196 which was used to evaluate the detection of small and large objects separately in [4]. It might be possible to set the threshold adaptively if an object size histogram is available. But knowing the object size histogram means we have fully labeled objects, implying we don't need a point dataset. Thus, the threshold is set to a fixed value for all experiments.

Method	Small P	Large P	All P	Small R	Large R	All R	All F-1
ScribbleNet [30]	0.0020	0.5650	0.4758	0.0074	0.6221	0.5041	0.4895
RU+w_{size} [4]	0.2593	0.9145	0.7941	0.3251	0.9338	0.8169	0.8053
Ours-I	0.3132	**0.9188**	0.8143	0.4551	**0.9379**	0.8452	0.8295
Ours-P-U	0.2702	0.8933	0.8111	0.6068	0.9124	0.8537	0.8319
Ours-P-USB	**0.3607**	0.9025	**0.8310**	**0.6161**	0.9286	**0.8686**	**0.8494**

Table 1. Main results. Average precision and recall for small objects, large objects, and all objects are reported. All F-1, average F-1 score for all objects are calculated by harmonic average of All P and All R. The original ScribbleNet [30] shows poor result because of the shallow back-bone network. We use ResNet-101 U-net for back-bone network. Addition of size weight used in building detection of an aerial image [4] increases the performance, but small objects are predicted as circular area (RU+w_{size}). Our method with the whole image and without point sampling shows better results (Ours-I). Considering uncertainty sampling only [15] increases detection rate, but also increases false positive rate (Ours-P-U). Our method with new point sampling rules shows better results (Ours-P-USB).

Method	Small P	Large P	All P	Small R	Large R	All R	All F-1
RU+w_{size} [4] (WHU)	0.4495	0.9094	0.8050	0.5503	0.9338	0.8373	0.8208
Ours-P-USB (WHU)	**0.4589**	**0.9318**	**0.8291**	**0.6277**	**0.9512**	**0.8698**	**0.8490**
RU+w_{size} [4] (Mass)	0.5256	0.3932	0.4816	0.6185	0.6083	0.6178	0.5413
Ours-P-USB (Mass)	**0.5470**	**0.4956**	**0.5149**	**0.6284**	**0.6633**	**0.6306**	**0.5669**

Table 2. Experimental results on WHU building dataset [13] and Massachusetts buildings dataset [19].

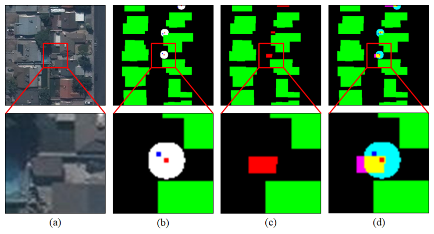

Figure 6. Imperfect point-labeled image with small radius ($r = 7$). (a) Input image. (b) Point label. (c) Full label. (d) Overlap of point label and full label. There occur 3 types of label change. The first one (yellow) and the second one (cyan) is changed from small object to unknown region and from background to unknown region, respectively. The third one (magenta) is erroneously changed from small object to background.

We also experiment with WHU building dataset [13] and Massachusetts buildings dataset [19]. WHU building dataset has 4,736 training images and 1,036 test images. The size of original images is 512×512. The images are randomly cropped to 300×300 and resized to 256×256. Massachusetts buildings dataset has 137 training images and 4 test images. The size of original images is 1500×1500. These images are cropped to 300×300 with regular interval of 150×150, which results in 11,097 training images and 324 test images. Cropped images are resized to 256×256. We use the same threshold $T_s = 196$.

4.2. Point dataset

To check the performance of our approach for the image with small-point and large-full labels that we are interested in, we first explain how human labeling is performed. Referring to Figure 1 (c) and Figure 2 (c), given the threshold T_s, human annotators visually distinguish small buildings approximately using their eyes. Then, they sample one point randomly inside a small building with a double-click, and a circle at the sampled point with pre-defined radius is automatically generated. Then, they sample another point, inside the circle, which is in the background outside the small building. This point labeling is much easier than the full contour labeling.

In our experiments, we generate a simulated point-label dataset from each of three datasets with full pixel labels as follows:

1) For a small building which is determined using a threshold T_s and full labels, one point is randomly selected from its inside.

2) For the selected small object point, we generate a circle with a radius r containing the small object.

3) An appropriate background point which is not inside large buildings is randomly sampled from each of the circle.

4) The pixels corresponding to large objects are excluded from each circle generated above.

In this way, we can build a ground truth label map which consists of the regions of large objects, the points and circles of small objects, and their remainders. These correspond to large object class, small object class, unknown region and background class, respectively as shown in Figure 6.

In our implementation, instead of single 1×1-size point, we use a 3×3-size blob by considering morphological erosion and dilation with a 3×3 mask.

We use the value of r=21 for the radius of the small ob-

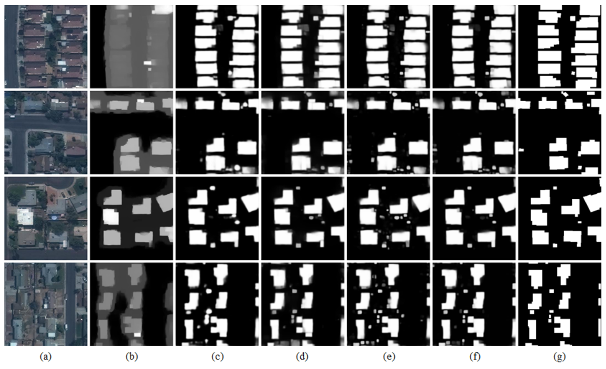

Figure 7. Prediction results of Table 1. (a) Input image. (b) ScribbleNet [30]. (c) RU+w_{size} [4]. (d) Ours-I. (e) Ours-P-U. (f) Ours-P-USB. (g) Ground truth.

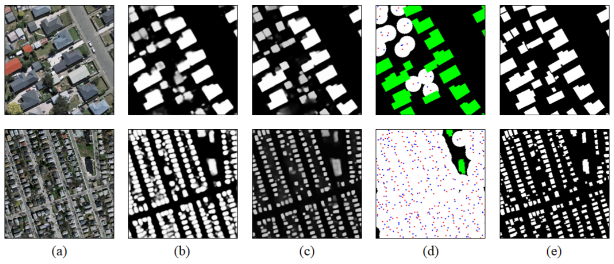

Figure 8. Prediction results of Table 2. (a) Input image. (b) RU+w_{size}. (c) Ours-P-USB. (d) Point label. (e) Full label.

ject mask. As shown in Figure 6, according to the value of radius r, some portion of a small object and background are labeled as unknown region. Also, some portion of small object can be mislabeled as background. Area of the label change in Figure 6(d) differs according to the value of r, and our point labeling scheme cannot be performed perfectly. From the ablation experiment on r, we can see that some reasonable range of r provides satisfactory performance although smaller values of r yield poor performance. Thus, there seems to be some degree of freedom in setting r, implying that background points can be selected easily and thus less time-consuming labeling is possible.

4.3. Main results

Table 1 compares our approach with two existing methods. ScribbleNet [30] is trained using the contours of large objects and the points of small objects as foreground scribble, and the remainder as background scribble. We can see that ScribbleNet does not work well for hybrid labeling.

We also compare U-net [24] based on ResNet-101 [11] with the size weight [4] (RU+w_{size}), which is used in building detection of aerial images, resulting the significantly improved performance. But it still does not well compared to our proposed model with whole image and without point sampling (Ours-I). The region separation used in the proposed model serves to emphasize small objects in the loss function, similarly to the size weight, and it also refines the shape of small objects.

The addition of uncertainty sampling (Ours-P-U) in a way similar to PointRend [15] increases detection rate, but it is vulnerable to false positive small objects, increasing the recall of small objects at the expense of decreased precision. However, our proposed method with new sampling rules (Ours-P-USB) reduces false positive rates, improving overall performance. Both the memory bank and the false positive sampling contribute to performance improvement. Detailed performance change according to point sampling rules, network architecture, memory architecture and small object prediction is shown in the supplementary material which shows robustness of our method to hyperparameter change.

We also compared RU+w_{size} with Ours-P-USB for WHU building dataset [13] and Massachusetts buildings dataset [19] in Table 2 and Figure 8, respectively. In WHU building dataset, the size of the small buildings tends to close to the threshold value, contrary to CrowdAI dataset. The performance improvement due to the proposed algorithm was similarly reflected into both small and large buildings because they are similar in size. In Massachusetts building dataset, there are many densely packed small buildings. So the cluster of small buildings can be misrecognized as a large building, which leads to poor performance on large buildings by the baseline method (RU+w_{size}).

It is noted that our approach yields performance similar to existing segmentation methods when it is trained with fully labeled data without using point labels.

5. Conclusion

We have proposed a weakly supervised segmentation network for small and large objects. Small buildings are labeled by using our new simple point labeling process whereas large buildings are fully labeled like other methods. We also proposed a small object mask to separate losses of small objects and large objects. To solve the problem of using small number of point labels, we use a memory bank to remember and update feature vectors of small object points during training, with sampling of uncertain and false positive data. Experimental results show the effectiveness of our approach. In future work, we plan to enable point labeling of large objects with adaptive unknown regions.

Acknowledgements. This work was conducted by Center for Applied Research in Artificial Intelligence (CARAI) grant funded by Defense Acquisition Program Administration (DAPA) and Agency for Defense Development (ADD) (UD190031RD).

References

[1] Bogdan Alexe, Thomas Deselaers, and Vittorio Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(11):2189–2202, 2012.

[2] Amy Bearman, Olga Russakovsky, Vittorio Ferrari, and Li Fei-Fei. What's the point: Semantic segmentation with point supervision. *Eur. Conf. Comput. Vis.*, pages 549–565, 2016.

[3] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3213–3223, 2016.

[4] Jakub Czakon, Kamil A. Kaczmarek, Andrzej Pyskir, and Piotr Tarasiewicz. Best practices for elegant experimentation in data science projects (case study). *EuroPython*, 2018.

[5] Rongsheng Dong, Xiaoquan Pan, and Fengying Li. Denseunet-based semantic segmentation of small objects in urban remote sensing images. *IEEE Access*, 7:65347–65356, 2019.

[6] Mark Everingham, Luc Van Gool, Christopher K. I. Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

[7] Chenyou Fan, Xiaofan Zhang, Shu Zhang, Wensheng Wang, Chi Zhang, and Heng Huang. Heterogeneous memory enhanced multimodal attention model for video question answering. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 1999–2007, 2019.

[8] Dong Gong, Lingqiao Liu, Vuong Le, Budhaditya Saha, Moussa Reda Mansour, Svetha Venkatesh, and Anton van den Hengel. Memorizing normality to detect anomaly: Memory-augmented deep autoencoder for unsupervised anomaly detection. *Int. Conf. Comput. Vis.*, pages 1705–1714, 2019.

[9] Ryuhei Hamaguchi and Shuhei Hikosaka. Building detection from satellite imagery using ensemble of size-specific detectors. *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 187–191, 2018.

[10] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9729–9738, 2020.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 770–778, 2016.

[12] Suyog Dutt Jain and Kristen Grauman. Predicting sufficient annotation strength for interactive foreground segmentation. *Int. Conf. Comput. Vis.*, pages 1313–1320, 2013.

[13] Shunping Ji, Shiqing Wei, and Meng Lu. Fully convolutional networks for multisource building extraction from an open aerial and satellite imagery data set. *IEEE Transactions on Geoscience and Remote Sensing*, 57(1):574–586, 2018.

[14] Michael Kampffmeyer, Arnt-Borre Salberg, and Robert Jenssen. Semantic segmentation of small objects and modeling of uncertainty in urban remote sensing images using deep convolutional neural networks. *IEEE Conf. Comput. Vis. Pattern Recog. Worksh.*, pages 1–9, 2016.

[15] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 9799–9808, 2020.

[16] Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vazquez, and Mark Schmidt. Instance segmentation with point supervision. *arXiv:1906.06392*, 2019.

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. *Eur. Conf. Comput. Vis.*, pages 740–755, 2014.

[18] Jinjie Mai, Meng Yang, and Wenfeng Luo. Erasing integrated learning: A simple yet effective approach for weakly supervised object localization. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 8766–8775, 2020.

[19] Volodymyr Mnih. Machine learning for aerial image labeling. 2013.

[20] Sharada Prasanna Mohanty. Crowdai mapping challenge 2018: Baseline with mask rcnn, 2018. https://github.com/crowdai/crowdai-mapping-challenge-mask-rcnn.

[21] Hyunjong Park, Jongyoun Noh, and Bumsub Ham. Learning memory-guided normality for anomaly detection. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 14372–14381, 2020.

[22] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 652–660, 2017.

[23] Andreas Robinson, Felix Jaremo Lawin, Martin Danelljan, Fahad Shahbaz Khan, and Michael Felsberg. Learning fast and robust target models for video object segmentation. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 7406–7415, 2020.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015.

[25] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. *International Conference on Machine Learning*, pages 1842–1850, 2016.

[26] Zhiqiang Shen, Mingyang Huang, Jianping Shi, Xiangyang Xue, and Thomas S. Huang. Towards instance-level image-to-image translation. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 3683–3692, 2019.

[27] Matteo Tomei, Marcella Cornia, Lorenzo Baraldi, and Rita Cucchiara. Art2real: Unfolding the reality of artworks via semantically-aware image-to-image translation. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5849–5859, 2019.

[28] Sherrie Wang, William Chen, Sang Michael Xie, George Azzari, and David B. Lobell. Weakly supervised deep learning for segmentation of remote sensing imagery. *Remote Sensing*, 12(2):207, 2020.

[29] Syed Waqas Zamir, Aditya Arora, Akshita Gupta, Salman Khan, Guolei Sun, Fahad Shahbaz Khan, Fan Zhu, Ling Shao, Gui-Song Xia, and Xiang Bai. isaid: A large-scale dataset for instance segmentation in aerial images. *IEEE*

Conf. Comput. Vis. Pattern Recog. Worksh., pages 28–37, 2019.

[30] Jing Zhang, Xin Yu, Aixuan Li, Peipei Song, Bowen Liu, and Yuchao Dai. Weakly-supervised salient object detection via scribble annotations. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 12546–12555, 2020.

[31] Minfeng Zhu, Pingbo Pan, Wei Chen, and Yi Yang. Dm-gan: Dynamic memory generative adversarial networks for text-to-image synthesis. *IEEE Conf. Comput. Vis. Pattern Recog.*, pages 5802–5810, 2019.

[32] Łukasz Kaiser, Ofir Nachum, Aurko Roy, and Samy Bengio. Learning to remember rare events. *Int. Conf. Learn. Represent.*, 2017.