Interactive Multi-Class Tiny-Object Detection

Chunggi Lee Seonwook Park Heon Song Jeongun Ryu
Sanghoon Kim Haejoon Kim Sérgio Pereira Donggeun Yoo
Lunit Inc.
{cglee, spark, heon.song, rjw0205, seiker, oceanjoon, sergio, dgyoo}@lunit.io

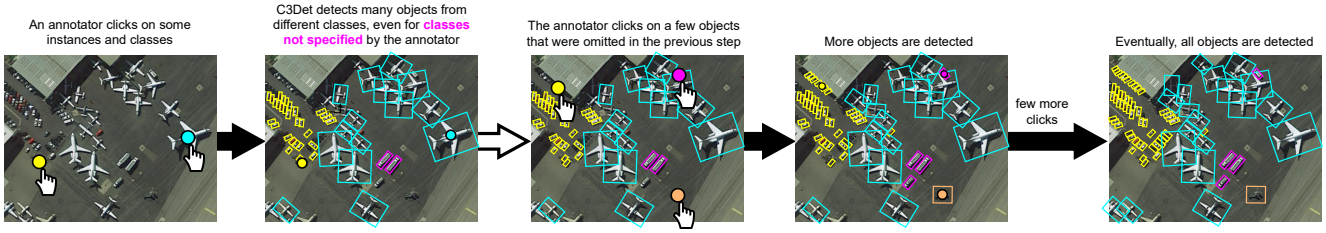


Figure 1. C3Det is a deep learning framework for interactive tiny-object detection that relates multiple annotator clicks to multiple instances and multiple classes of objects (each object class is depicted with a different color), in order to reduce overall annotation cost.

Abstract

Annotating tens or hundreds of tiny objects in a given image is laborious yet crucial for a multitude of Computer Vision tasks. Such imagery typically contains objects from various categories, yet the multi-class interactive annotation setting for the detection task has thus far been unexplored. To address these needs, we propose a novel interactive annotation method for multiple instances of tiny objects from multiple classes, based on a few point-based user inputs. Our approach, C3Det, relates the full image context with annotator inputs in a local and global manner via late-fusion and feature-correlation, respectively. We perform experiments on the Tiny-DOTA and LCell datasets using both two-stage and one-stage object detection architectures to verify the efficacy of our approach. Our approach outperforms existing approaches in interactive annotation, achieving higher mAP with fewer clicks. Furthermore, we validate the annotation efficiency of our approach in a user study where it is shown to be 2.85x faster and yield only 0.36x task load (NASA-TLX, lower is better) compared to manual annotation. The code is available at https://github.com/ChungYi347/Interactive-Multi-Class-Tiny-Object-Detection.

1. Introduction

Large-scale data and annotations are crucial for successful deep learning [22]. However in many real-world problems, annotations are very labor-intensive and expensive to acquire [8]. Annotation costs increase even higher when handling numerous tiny objects such as in remote sensing [7, 15, 33], extreme weather research [24], and microscope image analysis [12, 16]. These settings often require highly-skilled annotators and accordingly high compensation. For instance, cell annotation in Computational Pathology requires expert physicians (pathologists), whose training involves several years of clinical residency [3, 31]. Reducing cost and effort for these annotators would directly enable the collection of new large-scale tiny-object datasets, and contribute to higher model performances.

Several prior works have been proposed to reduce annotation cost in other tasks. Interactive segmentation methods [23, 35] focus on reducing the number of interactions in the segmentation of a *single* foreground object, which can be classified as a "many interactions to one instance" approach. However, tiny-objects annotation can benefit from a "many interactions to many instances" approach as one image can contain many instances. Object counting methods [4, 26] count multiple instances from a few user clicks and do follow a "many interactions to many instances" approach. However, these methods highlight only objects of the *same class* as the one being counted and thus can be classified as a "one class to one class" approach. However, images with tiny-objects are often composed of objects from multiple classes. Thus, tiny-object annotation should implement a "many classes to many classes" approach.

To address the above needs, we propose C3Det, an effective interactive annotation framework for tiny object detection. Fig. 1 shows how a user interacts with C3Det to create bounding-boxes of numerous tiny objects from multiple classes. Once a user clicks on a few objects and provides their class information, C3Det takes those as inputs

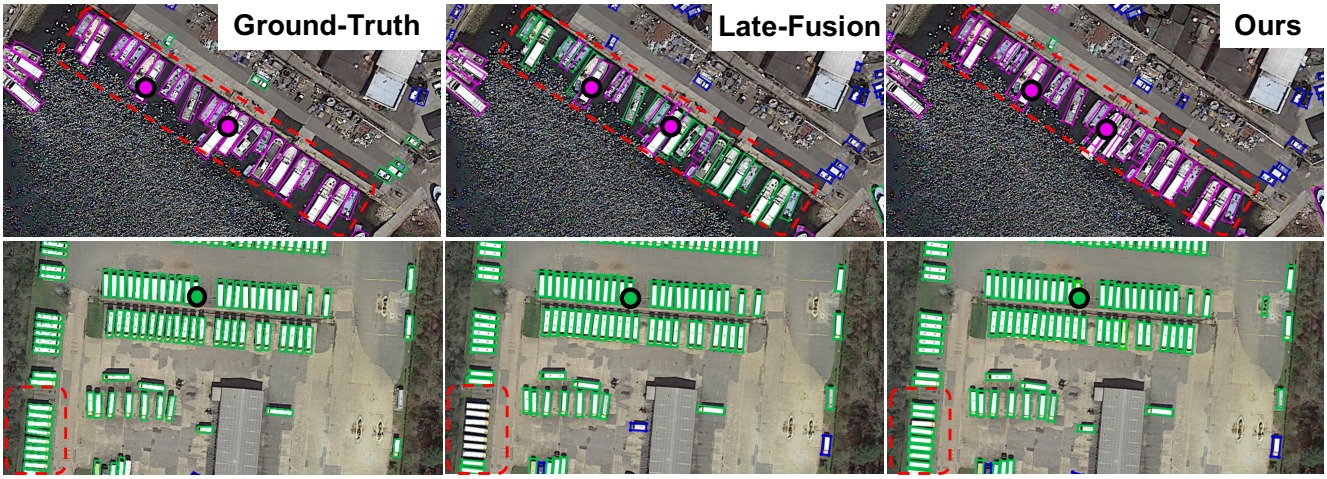


Figure 2. **The *global context* of user inputs matter.** "Late-Fusion" does not consider the global context and can miss far away objects (red dotted lines) from user inputs (marked as circles). C3Det captures the global context well and can detect far away objects.

and detects bounding boxes of many objects, even including object classes that the user did not specify. The user repeats this process until the annotation is complete. By utilizing user inputs in the "many interactions to many instances" and "many classes to many classes" way, C3Det can significantly speed-up annotation.

A key aspect of our approach is in making each user click influence objects that are nearby (local context) as well as far away (global context). To encourage the annotator-specified class to be consistent with model predictions, we insert user inputs (in heatmap form) at an intermediate stage in the model (late-fusion) and apply a class-consistency loss between user input and model predictions. This alone can capture local context well, but may miss far away objects. We therefore introduce the C3 (Class-wise Collated Correlation) module, a novel feature-correlation scheme that communicates local information to far away objects (see Fig. 2), allowing us to learn many-to-many instance-wise relations while retaining class information. Through extensive experiments, we show that these components combined, result in significant performance improvements.

To validate whether our performance improvements translate to lower annotation cost in the real-world, we perform a user study with 10 human annotators. Our approach, C3Det, when combined with further manual bounding box corrections, is shown to be 2.85× faster and yield only 0.36× task load (NASA-TLX) compared to manual annotation, achieving the same or even better annotation quality as measured against the ground-truth. This verifies that C3Det not only shows improvements in simulated experiments, but also reduces annotation cost in the real-world.

In summary, we make the following contributions: (a) we address the problem of multi-class and multi-instance interactive annotation of tiny objects, (b) we introduce a training data synthesis and an evaluation procedure for this setting, (c) we propose a novel architecture for interactive tiny-object detection that considers both local and global implications of provided user inputs, and finally (d) our experimental results and user study verify that our method reduces annotation cost while achieving high annotation quality.

2. Related Work

In this section, we discuss previous works that attempt to reduce annotation cost. The broad difference between our method and previous approaches are summarized in Tab. 1.

Interactive Object Detection The earliest work [36] in interactive detection incrementally trains Hough Forests over many image samples, gradually reducing false positives and negatives. This method is shown to be effective in annotating cell images and pedestrian images, where there are typically no more than 20 instances from a single class. While [36] adopts incremental learning over 5 or more images, our CNN-based model can be applied immediately to new samples, and handle many more objects.

Closely related are weakly-supervised object detectors (WSOD) that take point input [6,28] for establishing one-to-one correspondences between points and objects. However, this is different from our setting as these WSODs do not learn the many-to-many correspondences between points and objects, and do not yield interactive detectors.

Interactive Object Counting. This is similar to our tiny-object setting, since many tiny objects in an image are counted. An early work [4] learns a per-pixel ridge regression to adapt to user-provided point annotations, and counts instances of the specified object class. More recently, in [26], a few box annotations of a target class are forwarded as user inputs to the counting model. However, these counting works consider the instances of a sin-

Approach	# of classes to annotate	Class-to-class relation	# of instances to annotate	Interaction-to-instance relation	Outputs
Previous Interactive Detection [36]	1	1-to-1	many	many-to-many	bboxes
Interactive Counting [4, 26]	1	1-to-1	many	many-to-many	positions
Interactive Segmentation [23, 35]	1	1-to-1	1	many-to-1	contour
Ours	many	many-to-many	many	many-to-many	bboxes

Table 1. When considering the relationship between user interaction (typically the clicking of points) and annotated objects, we address the "many interactions to many instances" and "many classes to many classes" setting.

gle object class, while C3Det detects objects from multiple classes, including those not explicitly specified by user clicks. Also, our method estimates accurate bounding boxes of all objects, while [26] outputs so-called density maps.

Interactive Object Segmentation. In this setting, users mark a few points on an image to yield a segmentation of a single foreground object. The earliest methods [5] apply graph-cut, using an energy minimization method to separate foreground and background cues based on intensity changes, yielding optimal object boundaries. Instead of requiring many exemplar strokes to indicate foreground versus background, GrabCut [29] only requires a box to be drawn over the object. Learning-based methods are first introduced in [35], where users provide a few positive and negative clicks for object segmentation. The problem setting is formalized by introducing a training sample synthesis method, which is then followed by subsequent works [1,9,14,17,20,21,30,37]. C3Det is similar to these learning-based methods in that we also synthesize user inputs from bounding box annotations for training but differs in the two following ways. First, for every user input, C3Det annotates multiple objects from multiple classes at the same time, while these methods annotate a single object. Second, C3Det considers local-global relations to detect objects far away from the user's clicks, while interactive segmentation annotates just the object under the cursor.

3. Overview

Before we dive into describing the details of our method, we briefly motivate the broader decisions we made in building the C3Det framework. All of our decisions are based on a simple (yet important) goal: reducing the real-world annotation cost of tiny-object detection. This can be achieved by reducing annotation time and the number of interactions required from the annotator. Our C3Det framework addresses and improves on these aspects.

A quickly responding system improves both the annotation time and user experience. Taking inspiration from literature on deep interactive segmentation [35], we train a convolutional neural network (CNN) that only requires a simple feed-forward operation at test time. This has speed benefits compared to incremental learning approaches [36]. We later show in Sec. 6 that our CNN-based system reduces annotation time significantly when compared with a fully manual annotation method, by operating at interactive rates[1].

To further reduce the number of interactions required we devise two strategies. First, compared with bounding box inputs, we opt to receive the user inputs as point positions (via mouse click) along with object class. This allows annotators to simply click on a tiny object and specify its class, yet yield full bounding boxes in the outputs of C3Det. Second, we decide to preemptively detect objects from classes that have not been selected by the annotator yet. This allows the annotator to focus only on mistakes made by the annotation system.

Together with the contributions described in the following sections, we propose a meaningful solution to the problem setting of interactive multi-class tiny-object detection.

4. Method

In this section, we describe the proposed method. First, we introduce the overall architecture of C3Det. Next, we describe a training data synthesis procedure for multi-class and multi-instance interactive object detection. Finally, we describe each component of C3Det: the Late Fusion Module (LF), the Class-wise Collated Correlation Module (C3), and User-input Enforcing Loss (UEL).

4.1. Network Architecture

C3Det detects objects in a given image guided by a few user inputs, and outputs bounding boxes and the class of as many objects as possible, including those that are not specified by such inputs. We denote the input image as $\boldsymbol{I}$, and the number of user inputs as K.

Each user input is denoted as $\left(\boldsymbol{u}_k^{pos},\, u_k^{cls}\right)$, where k is the index of user input, $\boldsymbol{u}_k^{pos}$ defines a 2D position, and $u_k^{cls} \in \{1 \dots C\}$ is the object class. At inference time, $\left(\boldsymbol{u}_k^{pos},\, u_k^{cls}\right)$ is provided by a user, while at training and validation time, it follows the center point and class of the chosen ground-truth bounding box. Before passing the user inputs to the model, we convert each input $\left(\boldsymbol{u}_k^{pos},\, u_k^{cls}\right)$ as a heatmap $\boldsymbol{U}_k$ by placing a 2D Gaussian centered at $\boldsymbol{u}_k^{pos}$ with a predefined standard deviation σ_{heatmap}.

The input image $\boldsymbol{I}$ is first forwarded through a CNN feature extractor to yield a feature map $\boldsymbol{F}_{\text{I}}$. Separately, the user input heatmaps, $\boldsymbol{U}_{1\dots K}$, are passed to the LF and C3 modules, which utilize user inputs in local and global manners,

[1] C3Det responses takes just a few seconds on our user study GUI.

respectively. The outputs of these modules, $\boldsymbol{F}_{\mathrm{LF}}$ and $\boldsymbol{F}_{\mathrm{C3}}$, are then concatenated to $\boldsymbol{F}_{\mathrm{I}}$ before passing on to the next layers (see Fig. 3).

As C3Det only modifies the outputs of the backbone network, it is applicable to both one-stage and two-stage architectures. In the case of Faster R-CNN [27] and RetinaNet [19], for example, the concatenated outputs are passed on to the region proposal network (RPN) and to the classification and box regression subnets, respectively.

4.1.1 Training Data Synthesis

During training, we simulate the user inputs based on ground-truth annotations. First, we randomly sample a target number of user inputs from a uniform distribution $N_u \sim \mathcal{U}_{[0,20]}$. While we define the uniform distribution to extend to 20 only, this hyper-parameter can be adjusted as necessary. We then sample $K = \min(N_u, N_a)$ objects (without replacement) from the ground-truth, where N_a denotes the number of available objects for the current sample. The object centers and class indices are then passed on to C3Det as user inputs.

4.1.2 Late Fusion Module (LF)

When incorporating user input heatmaps to the network, two common approaches in interactive segmentation are early-fusion [14, 30, 34, 35] and late-fusion methods [2, 25, 37]. Early-fusion methods concatenate user-input heatmaps to the input image, while late-fusion methods inject user-input heatmaps to an intermediate layer in the network, with [37] or without [2, 25] processing the heatmaps with CNN layers. Prior insights show that late-fusion outperforms early-fusion [25, 37], and we find that this is also the case for interactive tiny-object detection.

To handle a varying number of user inputs, while maintaining the class information of the given inputs, we group the K user input heatmaps by class, then apply a pixel-wise $\max$ operation to each group to yield C heatmaps. For the case where no inputs are provided for an object class, we simply pass a heatmap filled with zeros. The heatmaps are passed to the LF module (a CNN-based feature extractor such as ResNet-18) that outputs feature maps $\boldsymbol{F}_{\mathrm{LF}}$.

The LF module handles these heatmaps without any global pooling, and therefore does not lose any spatial information. For the local area around a user input $\boldsymbol{u}_k^{pos}$, the predicted objects' class can be directly affected by the user input u_k^{cls}. We can therefore consider the LF module as one that considers the *local context* of user inputs.

4.1.3 Class-wise Collated Correlation Module (C3)

While understanding the local context can help in predicting the correct class for objects near to user inputs, objects far away from user inputs must be impacted in a different way. Recently, in [26], a correlation operation between $\boldsymbol{F}_{\mathrm{I}}$ and user input related features was used to improve object counting performance, using a few exemplars to count as many similar objects as possible in a given image. Similarly, we suggest to extract template features from $\boldsymbol{F}_{\mathrm{I}}$ based on user inputs, perform correlation with $\boldsymbol{F}_{\mathrm{I}}$ (see Fig. 4), and merge the correlation maps class-wise.

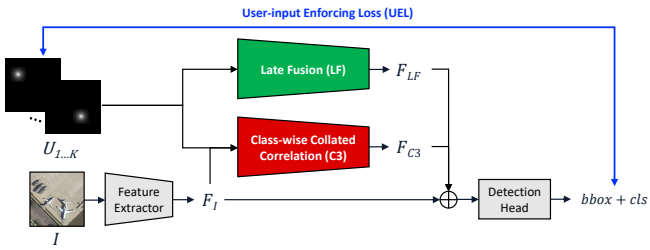


Figure 3. **Overall architecture.** User inputs are processed and considered at both local (by Late Fusion) and global (by Class-wise Collated Correlation) context scales for multi-class multi-instance interactive tiny-object detection. The "$\oplus$" symbol indicates channel-wise concatenation.

For each provided user input heatmap $\boldsymbol{U}_k$[2], we perform the following to obtain a "template" vector,

$$\boldsymbol{T}_k(i) = \sum_{x,y} \boldsymbol{F}_{\mathrm{I}}(i,x,y)\ \boldsymbol{U}_k(x,y), \tag{1}$$

where i refers to a channel index and x, y the column and row indices in $\boldsymbol{F}_{\mathrm{I}}$ and $\boldsymbol{U}_k$. This template vector can then be used as follows to generate a correlation map $\boldsymbol{M}_k$,

$$\boldsymbol{M}_k(x,y) = \sum_{i} \boldsymbol{T}_k(i)\ \boldsymbol{F}_{\mathrm{I}}(i,x,y). \tag{2}$$

Once K correlation maps are computed, we combine them class-wise based on u_k^{cls}, via an element-wise $\max$ operation as defined by,

$$\boldsymbol{F}_{\mathrm{C3}}(c,x,y) = \max\{\boldsymbol{M}_k(x,y)\,|u_k^{cls} = c, \forall k \in [K]\}, \tag{3}$$

where c refers to a class index. Classes that do not have any associated user inputs are simply represented by a correlation map filled with zeros.

This reduction allows us to produce C correlation maps to pass on to the next stages, no matter how many user inputs are provided. We describe this approach as a *correlate-then-collate* method, where each user input is handled independently. An intuitive alternative is a *collate-then-correlate* method, where user input heatmaps are combined by class first, to perform the correlation operation once per object class. The *collate-then-correlate* alternative may be more robust to the choice of user input, but also assumes

[2]The heatmap is typically resized to match the size of $\boldsymbol{F}_{\mathrm{I}}$ and normalized such that it sums to 1.

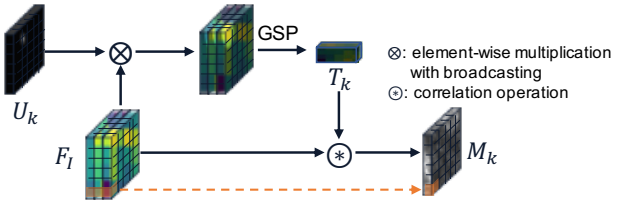


Figure 4. **Procedure of generating a correlation map.** A template vector is extracted from a feature map based on a user-input. The correlation map is computed between the template vector and the feature map. GSP stands for "global sum pooling".

that each object class can be described by a single feature representation. In our ablation study (see Fig. 8c), we show that the *correlate-then-collate* method performs better, and thus choose this to define our C3 module.

The explicit correlation operation performed by the C3 module allows for local features to be compared across the entire image. This extends the effect of user inputs on the model's predictions beyond the considerations of the LF module. In other words, we can consider the C3 module as one that considers the *global context* of user inputs by learning many-to-many instance-wise relations.

4.1.4 User-input Enforcing Loss (UEL)

When a user specifies an object to be of a certain class, C3Det should reflect this class on its prediction. Therefore, we propose to apply a training-time consistency loss between user inputs and model predictions through a User-input Enforcing Loss that enforces a class-wise consistency.

For each simulated user input, $(\boldsymbol{u}_k^{pos}, u_k^{cls})$, we retrieve the associated ground-truth bounding box $\boldsymbol{y}_k^{bbox}$. We compare each of these ground-truth objects with all J predicted objects (indexed by $j \in \{1 \dots J\}$). Each prediction consists of a bounding box $\hat{\boldsymbol{y}}_j^{bbox}$ and class $\hat{y}_j^{cls}$. To compute the loss, we check for a non-zero intersection-over-union (IoU) between every input-prediction pair, and apply a class-consistency loss. The full loss is formulated as,

$$\mathcal{L}_{\text{UEL}} = \sum_{j,\,k} \mathbb{1}_{IoU(\hat{\boldsymbol{y}}_j^{bbox}, \boldsymbol{y}_k^{bbox}) > 0} \cdot \ell(\hat{y}_j^{cls}, u_k^{cls}) \qquad (4)$$

where ℓ is a loss function such as the cross entropy loss or the focal loss, depending on the main task loss.

5. Experimental Results

To validate the proposed approach, we train and evaluate on two multi-class tiny-object datasets, Tiny-DOTA and LCell. We compare the performance of C3Det against several baseline methods, and show that C3Det applies to both one-stage and two-stage detectors such as RetinaNet and Faster R-CNN, respectively. These are standard baselines architectures for detecting oriented bounding boxes on the DOTA dataset [32, 33], and our method should apply to other object detection architectures as well[3]. Furthermore, we present ablation studies to verify the efficacy of our modules. For the implementation details of our model, please refer to our supplementary materials.

	Num. classes	Num. patches			Mean objects / patch		
		train	val	test	train	val	test
Tiny-DOTA	8	11198	1692	2823	38.5	42.8	35.3
LCell	8	3681	250	823	79.3	82.0	99.6

Table 2. Comparison of statistics between Tiny-DOTA and LCell. The number of patches reported for Tiny-DOTA are counted after subdividing the original images as described in Sec. 5.1. LCell contains a higher mean number of objects per patch.

5.1. Datasets

Tiny-DOTA The DOTA dataset [11, 33] consists of aerial images and includes a variety of tiny (e.g. vehicles and ships) and larger objects (e.g. soccer ball field and basketball court). Following [32] which deals with tiny object detection, we filter out the larger objects in the DOTA v2.0 dataset, yielding 8 tiny-object classes. Furthermore, our procedure requires frequent querying of the test-set ground-truth (for validation-time user-input synthesis), yet the original DOTA test set's labels are not publicly available. Hence, we split the original dataset into training, validation, and test subsets (70%/10%/20% split) for our experiments. We denote this dataset as Tiny-DOTA[4]. Following [33], we generate a series of 1024×1024 pixel patches from the revised dataset with a stride of 512 pixels, and train our models to detect oriented bounding boxes (OBB).

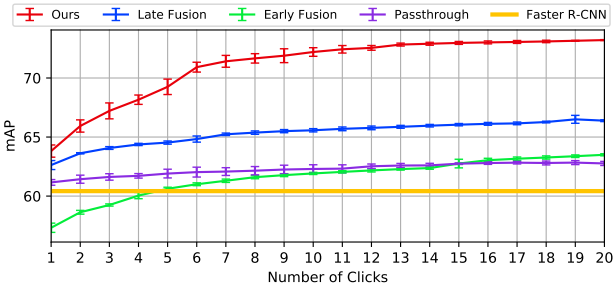
LCell LCell is a private breast cancer histopathology dataset with bounding box annotations for 8 cell classes. LCell consists of 768×768 size patches and has 3681 training, 250 validation, 823 test samples. We show in Tab. 2 that on average, the patches in LCell contain twice as many objects as in Tiny-DOTA. Further information about LCell is provided in our supplementary materials.

5.2. Evaluation Procedure

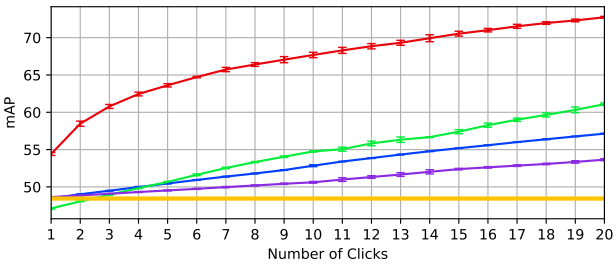
The evaluation of an interactive annotation system is a challenging topic. In the most ideal case, we could evaluate using a large number of human annotators over many data samples, but this is somewhat infeasible and certainly not reproducible. We thus take inspiration from the evaluation procedure in [35] for interactive segmentation, which plots

[3] See supplementary materials for results on RoI Transformer [10].

[4] To enable reproducibility and future comparison of results, the train-validation-test split of Tiny-DOTA is available at https://github.com/ChungYi347/Interactive-Multi-Class-Tiny-Object-Detection.



(a) Tiny-DOTA



(b) LCell

Figure 5. C3Det (Faster R-CNN w/ R50-FPN) performance on Tiny-DOTA and LCell datasets compared to baseline methods.

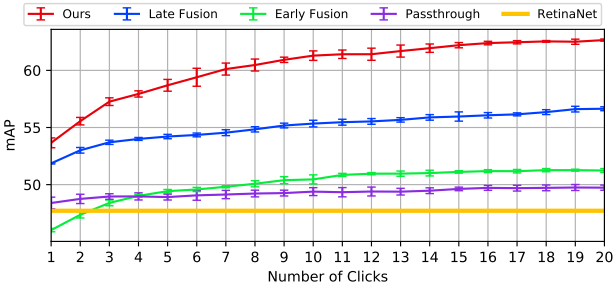


Figure 6. C3Det (RetinaNet w/ R50) performance on Tiny-DOTA compared to baseline methods.

task performance against simulated user clicks[5].

We simulate up to 20 "clicks" per image sample. This simulation over the full test set is an *evaluation session*. For each "click", we randomly sample an object from a given image's ground-truth (without replacement), taking the object's center position and class index as the simulated user input. When all available ground-truth objects are provided as simulated user inputs (for an image sample), no further user inputs are provided (similar to the training-time sampling method in Sec. 4.1.1). This results in a set of predicted bounding boxes for an increasing number of users' clicks. At each step, mAP[6] is computed over all test set predictions for the corresponding number of clicks, and a plot of mAP versus clicks can then be made. We perform five independent *evaluation sessions*, and show the means and standard deviations of each data point using error bars.

[5] [35] also proposes a "mean number-of-clicks" metric, but this must be computed per-sample and cannot be done for the mAP metric.

[6] We compute mAP with an IoU threshold of 0.5.

5.3. Comparison to Baselines

We compare our C3Det approach against a few baseline methods in Fig. 5. For the compared methods, where applicable, we employ a Faster R-CNN architecture with a ResNet-50 (with feature-pyramid network) feature extractor [18]. The lines labeled *Faster R-CNN* in Fig. 5 are the performance of the detector when simply trained on the labeled data without any interactive possibilities. We refer to this as a *baseline detector* in this section. The compared methods are as follows:

Ours. The full C3Det approach including the LF and C3 modules as well as the UEL loss.

Early Fusion. Early-fusion is a common method in interactive segmentation [14, 30, 34, 35] and thus we implement it by concatenating the user-inputs heatmaps to the input image before passing through the feature extractor. When drawing heatmaps, we use a larger σ_{heatmap} than other methods[7], as smaller Gaussians are less effective and their information can be lost in later layers.

Late Fusion. Late-fusion is also commonly used in interactive segmentation [2, 25, 37], and is a competitive baseline method. We implement this baseline by using our LF module but omitting the C3 module and UEL loss.

Passthrough. A naive yet effective baseline is one where the class value of user-inputs are simply applied to matching predicted bounding boxes from the baseline detector.

Results. We find that our proposed method out-performs all baselines consistently, quickly increasing in test set mAP with a few number of clicks, and reaching higher mAPs when the maximum number of clicks are provided. The Early Fusion and Late Fusion baseline methods out-perform the naive passthrough method, but by smaller amounts compared to our approach.

5.4. Application to a One-stage Detector

Our approach can apply to both two-stage and one-stage detector architectures. We show this by applying C3Det to the RetinaNet architecture (with ResNet-50 backbone) and evaluating on Tiny-DOTA. The one-stage results show similar tendencies as the two-stage case (Fig. 5), with the baselines showing modest improvements over the *baseline detector*, and our method showing large improvements. We thus show that our method can apply to both one-stage and two-stage architectures for object detection.

5.5. Varying Amount of Training Data

In a real-world scenario, one may question whether our approach applies to cases with lower number of training

[7] For evaluating on Tiny-DOTA, we choose $\sigma_{\text{heatmap}} = 9$ for "Early Fusion", and $\sigma_{\text{heatmap}} = 1$ for "Late Fusion" and "Ours".

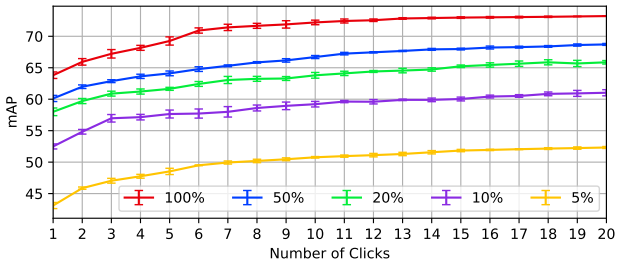


Figure 7. Decreasing the amount of training data (percentage of the full Tiny-DOTA training subset) still allows for C3Det to increase annotation quality with increasing number of clicks.

samples. We thus conduct an experiment by varying the amount of training data in Tiny-DOTA, using the Faster R-CNN architecture. Fig. 7 shows that our approach predicts bounding boxes with increasing mAP with increasing clicks, even with as little training data as 5% (only 559 samples). In the real-world, then, a small set of fully-annotated data could be collected initially in order to train C3Det. This could then be used to assist annotators in labeling additional samples. By repeating this process, even a large dataset could be annotated efficiently.

5.6. Ablation Studies

We conduct three ablation study on the Tiny-DOTA dataset to understand the impacts of our modules and loss. The evaluated method is C3Det based on the Faster R-CNN architecture with a ResNet-50 (with FPN) feature extractor.

User-input Enforcing Loss (UEL). Fig. 8a shows that the addition of the user-input enforcing loss results in significantly better performance (especially for no. of clicks > 3) compared to the case without UEL. It is clear that the UEL ensures that the consistency between user's inputs and the model predictions are improved. This is demonstrated by the overall better performance both in the presence of few clicks as well as in the case of many clicks. Furthermore, the smaller error bars (std. dev. of mAP over 5 trials) at high no. of clicks, indicate that applying the UEL allows the model to better understand and incorporate user inputs overall, without being too sensitive to which user inputs specifically are provided.

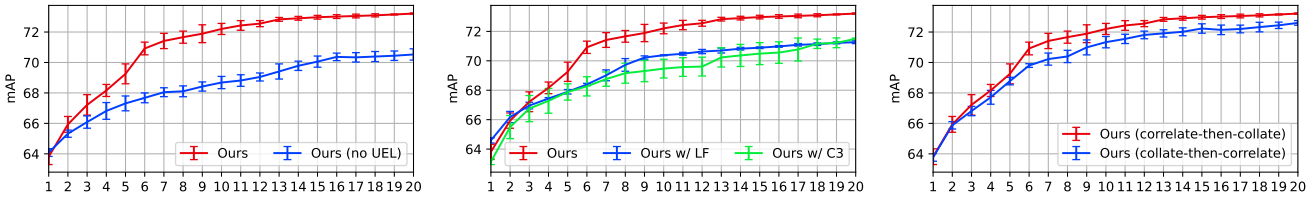
LF Module and C3 Module. In Fig. 8b, we train models with UEL and with either the LF module or C3 module, to compare the effect of the LF module against the effect of the C3 module. We find that the proposed LF module and C3 module on their own show good performance overall. However, it is when they are combined that a significant boost in performance is observed. We hypothesize that this is because the LF module allows for the model to better understand the implication of user inputs, in the local areas around the input positions. The C3 module on the other hand explicitly queries very far away objects for similarity. In a manner of speaking, the LF module helps the model understand the local context in relation to user inputs, and the C3 module helps the model understand the global context. This holistic approach is beneficial, as is evident by the large boost in performance.

Class-wise Feature Correlation. Our C3 module performs feature correlation per user input, then merges the correlation maps by class (correlate-then-collate). An alternative is to combine the user-input heatmaps class-wise first, then perform correlation (collate-then-correlate). The latter approach assumes that all objects of a specified class are represented by similar "template" features. In addition, it promises to be less sensitive to how well the user positions their input. In contrast, the chosen approach (C3) considers that objects from the same class can be represented by somewhat different features. By performing correlations for each user-input, the C3 module embraces the within-class diversity of objects. The results in Fig. 8c show that while both approaches work well, the correlate-then-collate method out-performs the collate-then-correlate alternative.

6. User Study

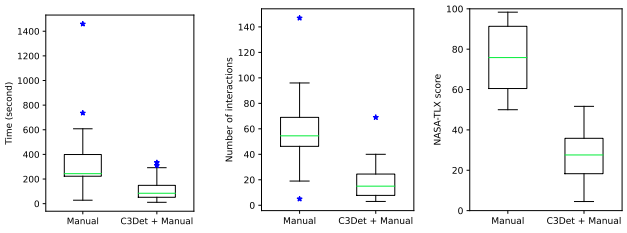
To evaluate the efficacy of C3Det in the real-world, we conduct a user study where annotators are asked to annotate OBB of objects on images taken from the Tiny-DOTA dataset. We sample 40 images from the test set for this task, which contain 10 to 100 objects. Our study is a within-subjects study, in which 10 participants perform their tasks with two conditions (in a random order). The two task conditions are: (a) fully-manual annotation and (b) interactive annotation using C3Det. In the fully-manual case, annotators select an object class, then make 4 mouse clicks per object to draw a quadrilateral that is oriented based on the object's orientation. In the interactive case, annotators are allowed to provide hints to C3Det by (a) selecting an object class, and (b) clicking on an example object. When the annotator is satisfied with C3Det's predictions, they are then allowed to revise mislabeled objects and add missing objects via manual annotation. Lastly, after completing each condition (manual or interactive), the annotators fill out the NASA-TLX questionnaire to assess their task load [13].

We analyze the annotation time and number of interaction for each condition, as shown in Fig. 9a and Fig. 9b. In the interactive case, users are allowed to further modify the bounding boxes predicted by C3Det, and so we call this the *C3Det + Manual* condition. The fully-manual alternative is simply called the *Manual* condition. The average annotating time spent for *C3Det + Manual* (114.7s) is 2.85 times lower than the *Manual* condition (327.73s). The number of interactions required for *C3Det + Manual* (17.93) is 3.25 times fewer than *Manual* (58.33), where interactions include the drawing and deleting of polygons, adding of user-



(a) User-input Enforcing Loss (UEL) (b) LF and C3 modules (c) Correlation/Collation Order

Figure 8. **Ablation study** of C3Det (Faster R-CNN w/ R50-FPN) on Tiny-DOTA. The graphs show the impacts of (a) User-input Enforcing Loss (UEL), (b) the combination of LF and C3 modules, and (c) the order of correlation and collation.



(a) Time (b) Interactions (c) NASA-TLX

Figure 9. Box plots of per-sample (a) time taken and (b) number of interactions, and overall (c) task load assessed using NASA-TLX.

inputs (for C3Det), changing the selected class, and final result submission. Please note that the time taken for *C3Det + Manual* includes model inference times, and shows that our method can work at interactive rates with annotators.

Fig. 9c shows the NASA-TLX [13] score for each task session. The median TLX score with the Manual approach is 75.83, and the score with C3Det is 27.58 (lower is better). A paired Wilcoxon signed-rank test with a continuity correction is conducted and we find that the difference in task workload is statistically significant at a significance level of 0.01 ($z' = -2.703$, $p' = 0.0069$, $r' = 0.604$). This means that annotating with C3Det takes less time, interaction, and workload as measured by NASA-TLX.

To evaluate the quality of the final annotations, we compute the mAP metric between the Tiny-DOTA ground-truth and the annotations acquired via our user study[8]. We additionally introduce the *C3Det Only* condition, which is the annotation acquired only via interacting with C3Det (without any manual modifications by the user). Fig. 10 shows the increase in mAP over time for the compared conditions. When considering how long it takes to achieve 67.9 mAP, the *Manual* condition takes 714.3s, while the *C3Det Only* and *C3Det + Manual* conditions take 294.2s and 144.2s respectively. This shows that *C3Det Only* allows for faster annotation than *Manual*, for comparable quality. Allowing further manual modifications in *C3Det + Manual* results in even better annotation quality, indicating that practitioners should consider allowing manual modifications even in interactive annotation systems. Considering that our user study participants are not expert annotators for Tiny-DOTA imagery, we believe that our results also indicate that *C3Det + Manual* can be a effective system for novices.

[8]Bounding box class-confidence scores are set to 1 to compute mAP.

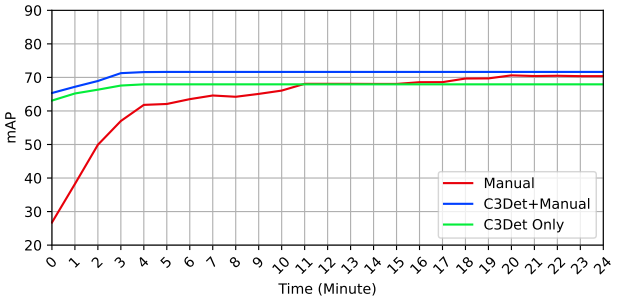


Figure 10. Annotation quality (mAP) versus annotation cost (time) for different annotation schemes. Our *C3Det + Manual* reaches 67.9 mAP five times faster than *Manual*.

7. Conclusion

We have shown that C3Det is a compelling approach for interactive multi-class tiny object detection. C3Det improves an annotation task which can otherwise be laborious and expensive. Our novel architecture considers the local and global implication of given user inputs in a holistic manner. A newly proposed training data simulation and an evaluation procedure for interactive multi-class tiny-object annotation defines a clear methodology for future work in this area. Our experimental results and user study verify that our C3Det outperforms existing approaches and can reduce cost while achieving high annotation quality. We hope that our approach alleviates concerns on annotation cost and workloads in the real-world (e.g., industry settings).

Limitations. In our work, we assume that annotators do not make mistakes when specifying object class. Therefore, C3Det may not be robust to rogue annotators. Separately, future work could propose an alternative to our point-based user inputs to target multiple objects with one interaction, further reducing annotation costs.

Potential Negative Impact. Our method improves annotation efficiency of imagery with many tiny objects. Unfortunately, surveillance imagery often contain many tiny objects and bad actors may benefit from our work. On the other hand, research on climate change, farm crop monitoring, and cancer research are highly beneficial to human society, hopefully offsetting concerns related to surveillance.

References

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, pages 859–868, 2018. 3

[2] Eirikur Agustsson, Jasper RR Uijlings, and Vittorio Ferrari. Interactive full image segmentation by considering all regions jointly. In *CVPR*, pages 11622–11631, 2019. 4, 6

[3] Mohamed Amgad, Lamees A Atteya, Hagar Hussein, Kareem Hosny Mohammed, Ehab Hafiz, Maha AT Elsebaie, Ahmed M Alhusseiny, Mohamed Atef AlMoslemany, Abdelmagid M Elmatboly, Philip A Pappalardo, et al. Nucls: A scalable crowdsourcing, deep learning approach and dataset for nucleus classification, localization and segmentation. *arXiv preprint arXiv:2102.09099*, 2021. 1

[4] Carlos Arteta, Victor Lempitsky, J Alison Noble, and Andrew Zisserman. Interactive object counting. In *ECCV*, pages 504–518. Springer, 2014. 1, 2, 3

[5] Yuri Y Boykov and M-P Jolly. Interactive graph cuts for optimal boundary & region segmentation of objects in nd images. In *ICCV*, volume 1, pages 105–112. IEEE, 2001. 3

[6] Liangyu Chen, Tong Yang, Xiangyu Zhang, Wei Zhang, and Jian Sun. Points As Queries: Weakly Semi-Supervised Object Detection by Points. In *CVPR*, 2021. 2

[7] Gong Cheng and Junwei Han. A survey on object detection in optical remote sensing images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:11–28, 2016. 1

[8] Michael Chui, James Manyika, and Mehdi Miremadi. What ai can and can't do (yet) for your business. *McKinsey Quarterly*, 1:97–108, 2018. 1

[9] Henghui Ding, Scott Cohen, Brian Price, and Xudong Jiang. Phraseclick: toward achieving flexible interactive segmentation by phrase and click. In *European Conference on Computer Vision*, pages 417–435. Springer, 2020. 3

[10] Jian Ding, Nan Xue, Yang Long, Gui-Song Xia, and Qikai Lu. Learning RoI Transformer for Detecting Oriented Objects in Aerial Images. In *CVPR*, 2019. 5

[11] Jian Ding, Nan Xue, Gui-Song Xia, Xiang Bai, Wen Yang, Micheal Ying Yang, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Object detection in aerial images: A large-scale benchmark and challenges, 2021. 5

[12] Simon Graham, Quoc Dang Vu, Shan E Ahmed Raza, Ayesha Azam, Yee Wah Tsang, Jin Tae Kwak, and Nasir Rajpoot. Hover-net: Simultaneous segmentation and classification of nuclei in multi-tissue histology images. *Medical Image Analysis*, 58:101563, 2019. 1

[13] Sandra G. Hart and Lowell E. Staveland. Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research. In *Advances in Psychology*, volume 52, pages 139–183. Elsevier, 1988. 7, 8

[14] Won-Dong Jang and Chang-Su Kim. Interactive image segmentation via backpropagating refinement scheme. In *CVPR*, pages 5297–5306, 2019. 3, 4, 6

[15] Ke Li, Gang Wan, Gong Cheng, Liqiu Meng, and Junwei Han. Object detection in optical remote sensing images: A survey and a new benchmark. *ISPRS Journal of Photogrammetry and Remote Sensing*, 159:296–307, 2020. 1

[16] Wenyuan Li, Jiayun Li, Karthik V Sarma, King Chung Ho, Shiwen Shen, Beatrice S Knudsen, Arkadiusz Gertych, and Corey W Arnold. Path r-cnn for prostate cancer diagnosis and gleason grading of histological images. *IEEE transactions on medical imaging*, 38(4):945–954, 2018. 1

[17] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, pages 577–585, 2018. 3

[18] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, pages 2117–2125, 2017. 6

[19] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, pages 2980–2988, 2017. 4

[20] Zheng Lin, Zhao Zhang, Lin-Zhuo Chen, Ming-Ming Cheng, and Shao-Ping Lu. Interactive image segmentation with first click attention. In *CVPR*, pages 13339–13348, 2020. 3

[21] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, pages 5257–5266, 2019. 3

[22] Dhruv Mahajan, Ross Girshick, Vignesh Ramanathan, Kaiming He, Manohar Paluri, Yixuan Li, Ashwin Bharambe, and Laurens Van Der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European conference on computer vision (ECCV)*, pages 181–196, 2018. 1

[23] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, pages 616–625, 2018. 1, 3

[24] Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Ebrahimi Kahou, Chris Pal, et al. Extremeweather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events. In *NeurIPS*, 2017. 1

[25] Kate Rakelly, Evan Shelhamer, Trevor Darrell, Alexei A Efros, and Sergey Levine. Few-shot segmentation propagation with guided networks. *arXiv preprint arXiv:1806.07373*, 2018. 4, 6

[26] Viresh Ranjan, Udbhav Sharma, Thu Nguyen, and Minh Hoai. Learning to count everything. In *CVPR*, pages 3394–3403, 2021. 1, 2, 3, 4

[27] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: towards real-time object detection with region proposal networks. *TPAMI*, 39(6):1137–1149, 2016. 4

[28] Zhongzheng Ren, Zhiding Yu, Xiaodong Yang, Ming-Yu Liu, Alexander G Schwing, and Jan Kautz. UFO^2: A Unified Framework Towards Omni-supervised Object Detection. In *ECCV*, 2020. 2

[29] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. " grabcut" interactive foreground extraction using iterated graph cuts. *TOG*, 23(3):309–314, 2004. 3

[30] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *CVPR*, pages 8623–8632, 2020. 3, 4, 6

[31] Jeroen van der Laak, Geert Litjens, and Francesco Ciompi. Deep learning in histopathology: the path to the clinic. *Nature medicine*, 27(5):775–784, 2021. 1

[32] Jinwang Wang, Wen Yang, Haowen Guo, Ruixiang Zhang, and Gui-Song Xia. Tiny object detection in aerial images. In *ICPR*, pages 3791–3798, 2020. 5

[33] Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. Dota: A large-scale dataset for object detection in aerial images. In *CVPR*, June 2018. 1, 5

[34] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas Huang. Deep grabcut for object selection. In *BMVC*. BMVA Press, 2017. 4, 6

[35] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, pages 373–381, 2016. 1, 3, 4, 5, 6

[36] Angela Yao, Juergen Gall, Christian Leistner, and Luc Van Gool. Interactive object detection. In *CVPR*, pages 3242–3249. IEEE, 2012. 2, 3

[37] Shiyin Zhang, Jun Hao Liew, Yunchao Wei, Shikui Wei, and Yao Zhao. Interactive object segmentation with inside-outside guidance. In *CVPR*, pages 12234–12244, 2020. 3, 4, 6