Scaling up instance annotation via label propagation

Dim P. Papadopoulos*
MIT CSAIL
dimpapa@mit.edu

Ethan Weber*
MIT CSAIL
ejweber@mit.edu

Antonio Torralba
MIT CSAIL
torralba@mit.edu

Abstract

Manually annotating object segmentation masks is very time-consuming. While interactive segmentation methods offer a more efficient alternative, they become unaffordable at a large scale because the cost grows linearly with the number of annotated masks. In this paper, we propose a highly efficient annotation scheme for building large datasets with object segmentation masks. At a large scale, images contain many object instances with similar appearance. We exploit these similarities by using hierarchical clustering on mask predictions made by a segmentation model. We propose a scheme that efficiently searches through the hierarchy of clusters and selects which clusters to annotate. Humans manually verify only a few masks per cluster, and the labels are propagated to the whole cluster. Through a large-scale experiment to populate 1M unlabeled images with object segmentation masks for 80 object classes, we show that (1) we obtain 1M object segmentation masks with an total annotation time of only 290 hours; (2) we reduce annotation time by 76× compared to manual annotation; (3) the segmentation quality of our masks is on par with those from manually annotated datasets. Code, data, and models are available online[1].

1. Introduction

The incredible rise in computer vision over the past decade has been propelled by the creation of datasets with multi-million annotated images such as ImageNet [45], Places [67] and Kinetics [21]. For deep learning models to continue improving with higher capacity, there is a continual need for more training data. It has been shown that the training data must grow exponentially to see linear improvements in the models [50, 52, 69].

Manual annotation for instance segmentation is expensive as it requires humans to draw a detailed outline around every object in an image [8, 13, 33, 46, 68]. For example, annotating COCO [33] required 80 seconds per mask,

[1] http://scaling-anno.csail.mit.edu
*Denotes equal contribution

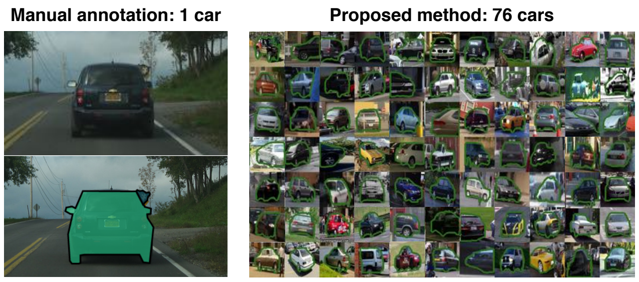

Figure 1. **Scaling up instance annotation.** We obtain 76 segmentation masks for the cost of manually drawing one.

an image in Cityscapes [8] took 1.5 hours, and annotating ADE20K [68] by a single annotator took several years. At this pace, constructing a dataset with 10M masks would require more than 200k hours and cost over $2M [33].

An alternative is interactive segmentation [1, 4, 7, 30, 34, 39, 43], where the human interaction is much faster (e.g., boxes, scribbles, clicks). Given this interaction, these methods infer the final object mask. They offer substantial gains in annotation time and can lead to larger datasets [4]. However, because annotators intervene on every object, the cost grows linearly with the number of annotations and therefore, at a larger scale, they become unaffordable.

In this paper, we propose a method for crowd-sourcing object segmentation masks that reduces the annotation time by almost two orders of magnitude (Fig. 1). At large scale, images contain many object instances with similar appearance. We exploit this by clustering mask predictions made by an instance segmentation model. Human annotators *verify* a few masks per cluster and we *propagate* the verification labels to the whole cluster. Since annotators interact with few masks per cluster, our cost scales with the number of clusters rather than masks.

Given a small initial image set with manually segmented objects and an annotation budget, our goal is to maximize the number of high-quality obtained annotations in an unlabeled set. We first train an instance segmentation model and obtain mask predictions on the unlabeled set and then hierarchically cluster class-specific masks. Starting from the root of the tree, we search for candidate clusters likely

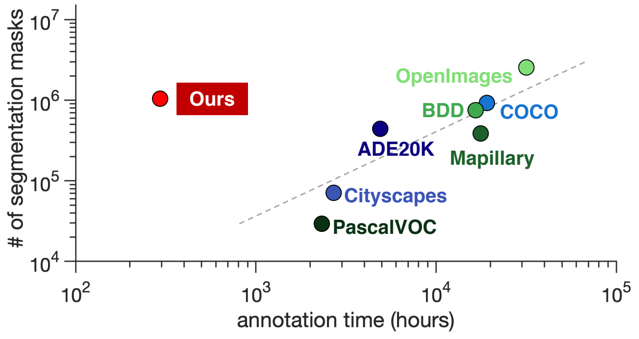

Figure 2. **Instance segmentation datasets and annotation cost.** The size vs. the total annotation time in hours of our annotations compared to the most popular instance segmentation datasets.

to contain high-quality masks. We ask human annotators to verify a few masks per cluster and we propagate the verification labels to the whole cluster. Once the search terminates, the new annotations are added to the initial pool.

We first conduct simulated experiments to explore the design space of our method. Then, we conduct a large-scale experiment to populate 1M unlabeled images from the Places dataset [67] with segmentation masks for 80 object classes. Our results show that we obtain 1M annotations with only 290 annotation hours, reducing the annotation time by 76× compared to manual annotation [33]. (Note that a higher annotation budget could lead to many more masks.) We also show that the mask quality is on par with those from manually annotated datasets, and our annotations leads to a better instance segmentation model than manual annotation given the same annotation time.

2. Related Work

Instance segmentation datasets are built by manually drawing accurate polygons around objects (e.g., PASCAL VOC [11], ADE20K [68], COCO [33], LVIS [13], Cityscapes [8], Mapillary [37] and BDD100K [64]). An exception is OpenImages V5 [4], where the masks are collected using interactive segmentation. Fig. 2 shows size and estimated annotation time for each dataset. Increasing a dataset by an order of magnitude is expensive as cost grows linearly with the number of annotations (e.g., scaling COCO by 10× would require 200k hours and $2M [33]).

Annotation cost and annotation time are proportional, so in this paper we use annotation time as it is a more objective measure than cost. The time to draw a mask or a polygon varies depending on the quality or the annotator's expertise [8, 11, 33, 64, 68]. As a reference time for manual annotation, we use the 80 s for drawing one polygon reported in [33]. Note that this is a conservative estimate because building a dataset requires overheads such as image classification and instance spotting [33]. For the total annotation time of our method, we always take into account the time of manually segmenting the initial training set and the time of verifying masks during our human-in-the-loop pipeline.

Interactive object segmentation started in the early 2000's with the seminal work of graph cut and iterative graph cuts [5, 6, 43]. After that, many approaches were proposed to reduce manual annotation effort using bounding boxes [9, 29, 59], point clicks [3, 4, 17, 30, 36, 40, 39, 60], scribbles [2, 3, 31], and editing polygons [1, 7, 34]. These methods offer remarkable gains in time compared to manual annotation. However, because they all require human intervention for every object, the annotation cost still grows linearly with the number of object segmentation masks.

Annotation propagation uses pre-segmented images to guide the segmentation of new ones by propagating the annotation signal [12, 18, 22, 27, 42, 44]. [18] selects which images should be annotated or used for propagation, but it focuses on large objects and the task of foreground-background segmentation. [22] uses human verification to propagate 3D shape part annotations on synthetic shape models. Here we present an active framework for instance annotation and propagation of large-scale image collections, where images contain multiple objects per image.

Group-based labeling assigns a single labeled image to a whole group [15, 51, 58]. In [65], human annotators label only a few images, which are used to train binary classifiers and automatically label many images at once with high confidence score. In Sec. 5.2 we modify [65] to work for instance segmentation and compare it with our method.

Active learning has been used in computer vision for the tasks of image classification [19, 20, 26, 35, 48, 62], object detection [55, 61] and semantic segmentation [18, 47, 53, 54]. The main goal of active learning is to maximize the model's performance on a test set while minimizing the number of provided human labels. Related to this goal, we aim to maximize the number of obtained annotations while minimizing the human annotation effort.

3. Overview

Given a small set of manually annotated images with segmentation masks and a large set of unlabeled images, our goal is to populate the unlabeled set with high-quality masks using as little human intervention as possible. Our pipeline consists of five steps (Fig. 3): (a) Detection: we deploy an instance segmentation model on an unlabeled set to obtain segmentation masks. (b) Clustering: class-specific masks are hierarchically clustered to obtain a tree. (c) Selecting clusters: we efficiently search the tree and select candidate clusters that are likely to contain high-quality masks. (d) Human annotation: for each candidate cluster, we sample a few masks and ask annotators to verify whether they are correct or not. We use these labels to estimate the quality of the clusters. (e) Propagation: If the estimated quality is very high or very low (i.e, the cluster almost exclusively contains

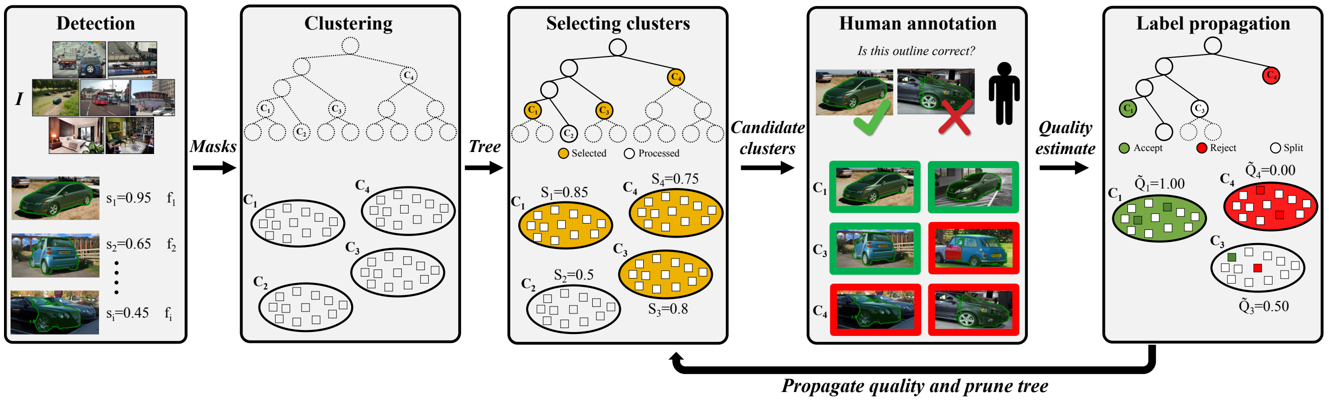

Figure 3. **The proposed human-in-the-loop pipeline.** We apply an instance segmentation model on an unlabeled set of images to detect object segmentation masks (***Detection***). Then, we use hierarchical clustering to cluster masks and form a tree of clusters (***Clustering***). We efficiently search the tree and select candidate clusters that are likely to contain high-quality masks (***Selecting clusters***). Human annotators verify a few masks per cluster (***Human annotation***) and we propagate the verification labels to the whole cluster (***Propagation***).

correct or wrong masks), we propagate the verification labels to the whole cluster and set it as a leaf. Otherwise, we further split it. We repeat (c), (d) and (e) to discover high-quality clusters with as few questions as possible.

4. Method

In this section, we present our pipeline to efficiently populate unlabeled images with object segmentation masks.

Detection. In this step, we train an instance segmentation model on a small initial set of manually segmented images to obtain object segmentation masks M on the unlabeled set of images I. For the instance segmentation model, we use PointRend [25]. Inspired by [16], we modify PointRend by adding a head trained to regress the Intersection-over-Union (IoU) of the output mask using the relaxed IoU loss [30]. We multiply this IoU score with the classification score of the box head to obtain a predicted score s for every predicted mask m. Therefore, for every m, we get a score s and a deep feature representation f. f and s together encode both the visual appearance and the mask quality.

Clustering. In this step, we cluster M using a feature representation $\mathcal{F}$. In Sec. 5.1, we evaluate different choices for $\mathcal{F}$. For each object class, we use bottom-up hierarchical agglomerative clustering (HAC) with the complete linkage function to obtain a binary tree T with clusters C as vertices. A cluster C_j is the set of vertices formed by the subtree with parent node C_j. The root of the tree C_r contains all masks M_r while the leaves are singleton clusters, i.e., $C_j = \{m_j\}$ for $j \in [1, |M_r|]$. After obtaining T, clusters are typically defined by setting a universal threshold at a certain height in the tree. However, the clusters at the same height are not equally pure (Fig. 5). For this reason, in the following steps we efficiently search the tree, actively select clusters, and query annotators to find the final cluster set.

Selecting clusters. In this step, we search T starting from the root node C_r to efficiently discover a set of high-quality clusters C^a. We use a priority queue to guide the search algorithm towards clusters that are likely to be of high quality. Different tree search algorithms lead to different cluster orderings, which results in a trade-off between cost and number of masks obtained during searching. However, all algorithms eventually visit the same clusters by the end of the procedure. The majority of the clusters in T are not high-quality and blindly annotating everything leads to an inferior trade-off, so instead we actively select which clusters to annotate. To do this, we estimate the likelihood of a cluster to be high-quality given the scores of the masks it contains.

A mask m_i is considered correct when its IoU with a perfect mask m_i^* capturing the object is greater than K_{iou}. We define the quality of m_i as $q_i = \mathbb{1}_{\text{IoU}(m_i) \geq K_{iou}}$. The quality Q_j of C_j is defined as $Q_j = \frac{1}{|C_j|}\sum_{q_i \in C_j} q_i$. We consider C_j to be of high quality when $Q_j \geq K_a$ and of low quality when $Q_j \leq 1 - K_a$.

We also define the score S_j of cluster C_j as the mean of the scores s of the masks it contains. Therefore, the likelihood of C_j to be of high quality given S_j is given by $P_h = P(Q_j \geq K_a | S_j)$. Similarly, $P_l = P(Q_j \leq 1 - K_a | S_j)$ is the likelihood that C_j is of low quality. We use S_j to learn the likelihoods P_h and P_l on a held-out set of images with ground-truths. We denote a threshold K_{pa} on S_j such that $P(Q_j \geq K_a | S_j < K_{pa}) \approx 0$. When $S_j > K_{pa}$, the cluster is annotated, otherwise it is unlikely to be of high quality and is split to its children (Fig. 4). Alg. 1 outlines this procedure.

Human annotation. In this step, we ask human annotators to quickly verify whether or not a displayed mask on an object is correct or not. This verification step is commonly used in efficient annotation schemes as each question takes less than 2 s [28, 41]. Our key difference is that for each

Algorithm 1: Selecting and annotating clusters.

```
Input: Binary tree $T$ with clusters $C$
Output: Accepted clusters $C^a = \{C_j | \tilde{Q}_j \geq K_a\}$
Accepted clusters $C^a = \{\}$ , Candidate clusters $C^c = \{C_r\}$
while $|C^c| > 0$ do
    Sort $C^c$ by scores $S$ and Pop $C_j$ from $C^c$
    Check $S_j$ to possibly split early
    Estimate quality $\tilde{Q}_j$ of $C_j$ // annotate
    if $\tilde{Q}_j \geq K_a$ then
        $C^a = C^a \cup C_j$ // accept
    else if $\tilde{Q}_j \leq 1 - K_a$ then
        // reject
    else
        Children $\{C_{left}, C_{right}\} = C_j$ // split
        $C^c = C^c \cup \{C_{left}, C_{right}\}$
return $C^a$
```

candidate cluster the annotator only verifies N_s randomly sampled masks and not all of them. The annotators are instructed to respond positively ($l_i = 1$) if the mask m_i correctly outlines the target object, and negatively otherwise ($l_i = 0$). We use the human responses of the N_s sampled masks $\hat{M}$ to obtain an estimated quality $\tilde{Q}$ for a cluster as $\tilde{Q} = \frac{1}{N_s} \sum_{l \in \hat{M}} l$.

Label propagation. In this step, we use $\tilde{Q}$ to propagate the human verification labels l to the rest of the masks in the cluster. If $\tilde{Q}_j \geq K_a$, we *accept* the cluster and propagate the positive verification labels ($l_i = 1$) to all the masks it contains. Similarly if $\tilde{Q}_j \leq 1 - K_a$, we *reject* the cluster and we propagate the negative verification labels ($l_i = 0$) to all the masks it contains. Otherwise, we further *split* the cluster into its children. The accepted and rejected clusters become leaf clusters in T by pruning all their children. The masks from the accepted clusters C^a form the output annotations, while the masks from the rejected clusters are discarded.

5. Simulated experiments

In this section, we perform simulated experiments to explore different design choices for each step of our pipeline and find parameters to minimize the annotation cost given a desired annotation quality. In Sec. 5.1, we perform experiments on ADE20K [68] while in Sec. 5.2 we simulate a large-scale scenario to estimate parameters not possible to estimate within a small dataset. In Sec 5.3, we perform experiments on COCO [33] and OpenImages [4].

Implementation details. For all experiments, we use PointRend [25] with FPN [32] and ResNet50 [14] as our instance segmentation model. During inference, we keep all predictions with a confidence score above 0.2.

5.1. Simulated experiments on ADE

Dataset. We use the training set of ADE20K [68], which consists of 20,210 images and ground-truth annotations for

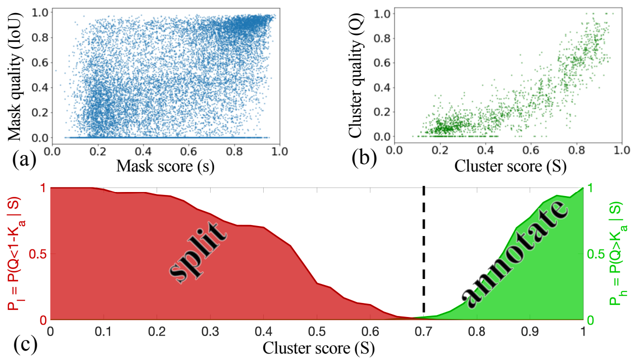

Figure 4. **Selecting candidate clusters.** (a) The real mask IoU as a function of the mask score s. (b) The real cluster quality (Q) as a function of the cluster score S. We observe much stronger correlation at the cluster level. (c) The estimated likelihood probabilities P_h and P_l given S. We only annotate clusters when $P_h > 0$. Otherwise, the cluster is split without any human intervention.

2,693 classes. We select the 80 most frequent foreground object ("thing") classes from the scene parsing ADE20K Benchmark [68]. For most experiments in this section, we use 10,105 images to train the instance segmentation model and consider the other half as the unlabeled pool. We study the effect of smaller initial sets an the end of this section.

Evaluation protocol. Our goal is to populate unlabeled images with high quality masks while minimizing human annotation effort. For this reason, we evaluate the trade-off between the quantity and quality of the obtained annotations versus the annotation cost. We measure *annotation quantity* as the total number of the obtained annotations. We measure *annotation quality* as the mean IoU of the obtained annotations with respect to the ground-truth ones. We compute each metric per class and report the sum for quantity and mean for quality over all classes. We measure *annotation cost* as the total number of annotated clusters. For all experiments in this section, we factor out the sampling step and when we annotate a cluster, we assume $\tilde{Q} = Q$. We consider masks with IoU ≥ 0.75 with the ground-truth as correct and we set $K_a = 0.85$.

Clustering. We first evaluate the result of the clustering step by examining how well different $\mathcal{F}$ construct T (Fig. 6 (a)). We examine four feature types. (a) The *Mask logits* of PointRend after passing them through a sigmoid; (b) The *Backbone features* of PointRend after ROI align; (c) The *MaskIoU features* from the second last layer of the mask IoU head. (d) The *Mask attention features* where we multiply the sigmoided mask logits with the backbone features. We obtain different numbers of clusters from each T by cutting at different distance thresholds in the HAC dendogram. In Fig. 6(a), we observe that the mask attention (orange line) achieves a better trade-off than the other feature types. Combining it with the mask score s (black line) leads to better clustering (more annotations given the same num-

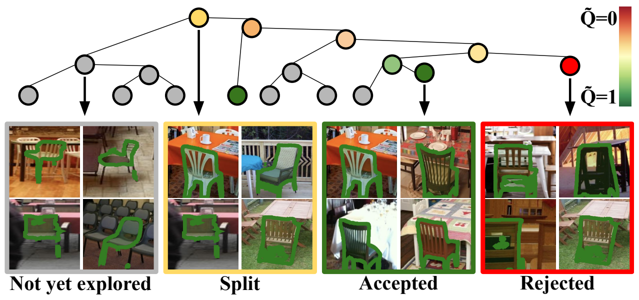

Figure 5. **Examples of clusters in tree of the chair class.** Clusters are *split* when containing both low and high quality masks, *accepted* when mostly correct, and *rejected* otherwise. The gray clusters have yet to be explored and have no quality estimate $\tilde{Q}$.

ber of clusters); we use this in the remainder of the paper. We also notice that the annotation quality (Fig. 6(a)) is always greater than 0.85. Even with random clustering, our verification and propagation guarantee high quality.

Searching the tree. We evaluate the *universal* thresholding under different number of clusters and four main search tree algorithms to search and annotate the tree: (1) *BFS*: breadth-first search; (2) *DFS*: depth-first search; (3) *DFS with heuristics*: depth-first search with a heuristic score to prioritize child clusters that are more likely to be of high quality; (4) *Heuristics only*: sort all candidate clusters according to the heuristic in decreasing order while searching the tree. For a fair comparison, we factor out the cluster selection and we select all the clusters for annotation.

Most of the search algorithms achieve a much better trade-off than the universal thresholding (blue line) (Fig. 6(b)). *Heuristics only: Mask score* achieves the best trade off (black line) and *DFS* alone achieves the worst as it blindly explores a path from the root until a random leaf (cyan line). Interestingly, using the mask score s as a heuristic metric we perform only slightly worse than the upper bound of using the real mean IoU of each cluster (dashed lines). We use the *Heuristics only: Mask score* as our searching algorithm for the remainder of this work.

Selecting clusters. The previous experiments assume every cluster is selected for verification. Here we evaluate the active selection of these clusters by following our selection mechanism described in Sec. 4 using different values for K_{pa}. In Fig. 6(c), we observe that very high values of K_{pa} result in a low number of good annotations (orange line), while low values have only a very small effect to the performance (cyan line). For the remainder of this work, we set $K_{pa} = 0.7$, which achieves the best trade-off (black line).

Initial training set. Here we evaluate the effect of the size of the initial training set. In Fig. 7(left), we show the annotation quantity versus the number of annotated clusters using 500, 1,000, 5,000 and 10,000 images as our initial set. We observe that with a larger set, accepted clusters contain more masks because the initial models are better. However, even with a very small set (e.g. 500 images), we obtain a significant boost in annotation quantity compared to manual annotation. Fig. 7(right) shows the annotation quantity versus the total time including the time to manually annotate the initial set. We assume 80 s to annotate each object in the initial set and 30 s to verify each cluster. Given a fixed budget, one should select the training set that reaches the corresponding vertical point of the dotted line to maximize the annotation quantity. We use 1,000 images as our initial set for the remainder of this work as it provides a good compromise between cost and maximum annotation quantity.

Comparison to confidence-based baseline. We compare our annotation quality and quantity against a simple baseline that selects masks based only on the PointRend confidence score. Our pipeline with a small annotation cost obtains 15,628 annotations with 0.83 quality. Using only the confidence score, we obtain only the top 67 high-confident annotations in order to maintain the same quality. From another perspective, the quality of the top 15,628 predictions is only 0.66. This shows that our method is doing much more than simply filtering out low confidence predictions.

5.2. Large-scale simulated experiments

In this section, we simulate a realistic large-scale scenario to estimate the parameters for the annotation and the propagation steps which cannot be optimized directly in ADE20K due to its small size. We estimate the number of verified samples per cluster N_s, the cluster quality threshold K_a, and the mask IoU threshold K_{iou}.

Evaluation set. The number of ground-truth masks per class in ADE20K varies from a few thousand to *only* a few hundred, and our experiments in Sec. 5.1 show that for most classes, the highly pure clusters contain on average less than five examples. For this reason, we design a simulated scenario under the assumption that at a larger scale the detection masks follow a similar joint distribution $P(f, s, IoU(m))$ (Recall that f is a feature representation vector, while s and $IoU(m)$ are scalars). We learn P on ADE20K in a class-specific way using a Gaussian Mixture model. Then, we sample triplets $\{f_i, s_i, IoU(m_i)\}$ and form simulated class-specific sets by following the class distribution and the number of instances per image of ADE20K. This leads to 100M triplets to run our pipeline.

Annotation and propagation. Following human annotation consistency [4, 13, 24, 68], a high quality instance segmentation dataset typically has a segmentation quality SQ between 0.8 and 0.85. SQ is defined as the average IoU for all matched manual annotations. We want to find the optimal values for N_s, K_a and K_{iou} while keeping $SQ \geq 0.85$. The higher the N_s, the more accurate the $\tilde{Q}$ is and the less noise the final dataset contains. However, N_s directly affects the annotation cost. Setting K_a or K_{iou} to 1 would lead to high quality; this, however, would result to a very

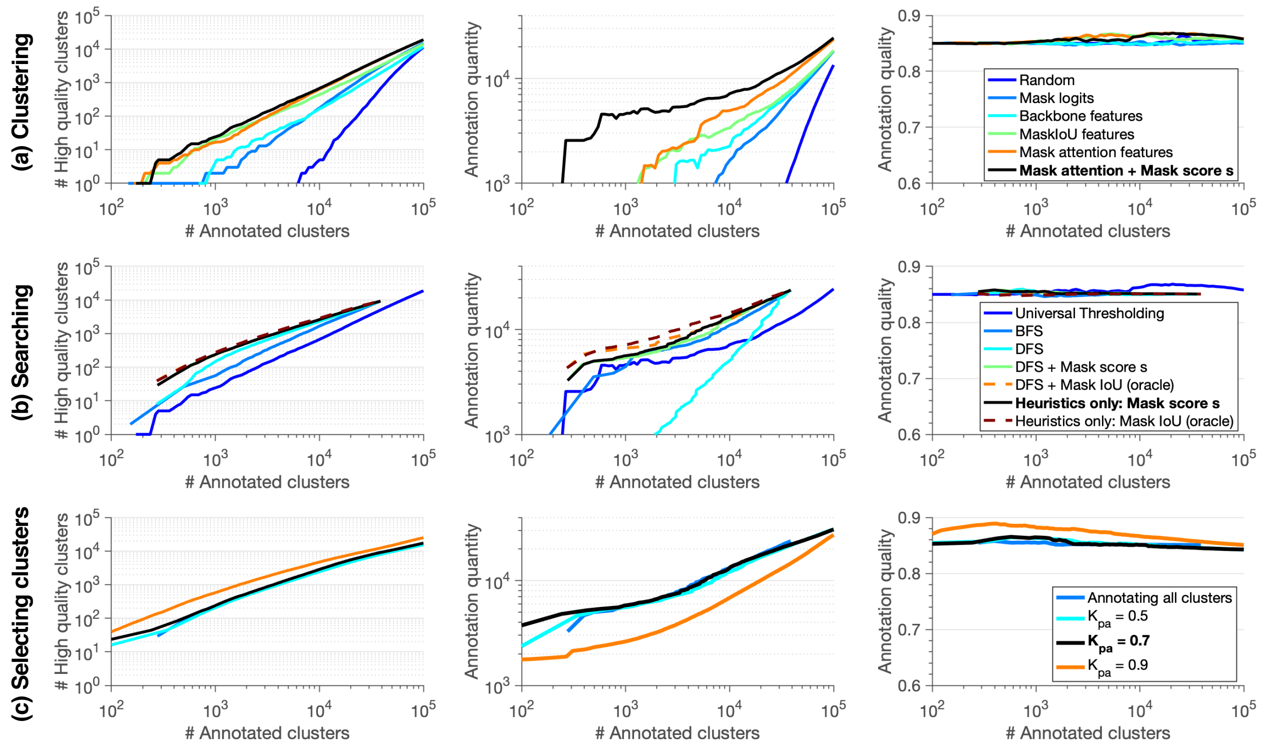

Figure 6. **Experimental results on ADE20K.** We show the number of high-quality clusters, the annotation quantity and the annotation quality versus the number of annotated clusters in log-log scale. **(a) Clustering**: The effect of using different feature representations $\mathcal{F}$ to construct T. **(b) Searching**: The effect of using different search algorithms when searching the tree T. **(c) Selecting clusters**: The effect of actively selecting clusters to annotate using different K_{pa} values. The black lines correspond to the best result in each row.

low number of annotations. We run our pipeline for different parameter choices and aim to keep $SQ \geq 0.85$. We first minimize N_s and then search over combinations of K_a and K_{iou} to find which values lead to the largest number of annotations. This results in $N_s = 15$, $K_a = 0.85$, and $K_{iou} = 0.75$. We note that if $N_s = \infty$, then $SQ = 0.89$.

Comparison to other verification approaches. We compare our pipeline to other verification-based approaches. In Fig. 8, we report the trade-off between annotation quantity and quality vs. the time in hours assuming 2 s for each binary question and 80 s for each manual drawing [33]. Our framework leads to a speed up of one to two orders of magnitude comparing to [33], while maintaining a high quality. We modified the method used for LSUN for instance segmentation [65], which iteratively (a) verifies a few images, (b) trains a binary classifier, and (c) applies it on unlabeled images to propagate labels and select ambiguous examples to annotate. Instead of an image pool, we use the pool of predicted masks from our initial model and apply the pipeline of [65] at masks without further changes. Our method achieves a better trade-off while maintaining a much better quality. Even in the object class bed, where our pipeline achieves the smallest improvement over brute-force verification [41], [65] leads to slightly more annotations but with low quality. For a fair comparison to manual annotation, we take into account the cost of the initial training set for all three verification approaches.

5.3. Experiments on COCO and OpenImages

Here we conduct simulated experiments on COCO [33] and OpenImages [4] (Fig. 9). We follow the setting of Sec. 5.1 and 5.2 exactly without tuning any parameters and train our initial segmentation model on 2k COCO images (80 classes). We run our pipeline on two unlabeled sets of images: (a) the remaining 121k COCO images, and (b) the OpenImages subset with a GT in the 60 COCO classes that overlap with the 350 OpenImages ones (316k images).

Quantity and annotation time. We obtain (a) 86k COCO and (b) 270k OpenImages annotations with only 12 h (+300 h to annotate the initial set). With 312 annotation hours, we obtain 14k instances with [33] and 28k with [4] in either dataset (the two lines in Fig. 9 appear horizontal because we manually annotate only 540 masks in 12 h [33]). Our time saving on OpenImages is 19× with respect to [33].

Quality. The quality of our masks is 84.6% on COCO. [4] reports 82.0% for [33] and 84.0% for [4] on a subset of COCO annotated with free-painting annotations. On OpenImages, we obtain 85.0% on the validation and test set vs

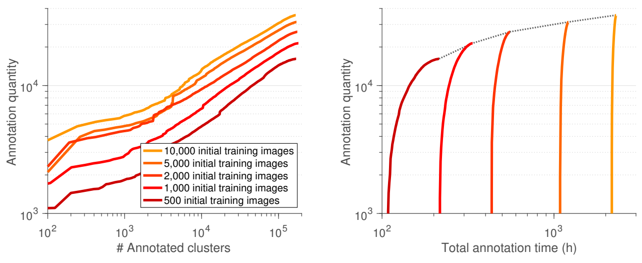

Figure 7. **Initial training set.** (Left) The annotation quantity vs. the annotated clusters for four initial annotated sets. (Right) The annotation quantity vs. the total annotation time in hours by taking into account the time to annotate the initial set.

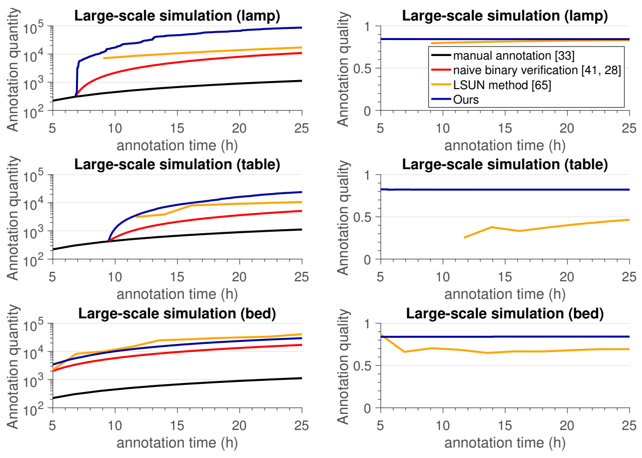

Figure 8. **Large-scale simulated experiment.** The annotation quantity and quality vs. the annotation time in hours for 3 object classes. We compare our pipeline with manual annotation [33], naive binary verification [28, 41] and the LSUN method [65].

86.0% with [4] (provided by the authors of [4]).

Hyperparameters. The annotation gain of our propagation method on COCO and OpenImages with the same hyperparameters tuned on ADE20K indicate that our method is not sensitive to per-dataset hyperparameter tuning.

6. Large scale experiment on Places

In this section, we present our results on a large-scale experiment to obtain object segmentation masks using our framework with a fixed annotation budget.

Data. We use 1M images from the trainval set of the Places dataset [67] as our unlabeled set and using our method we populate it with high-quality masks for 80 object classes. Unlike the simulated experiments (Sec. 5.1) where the unlabeled pool is limited, using a pool of 1M images allows us to demonstrate the scaling of the annotation process and the gain of our clustering approach at a large scale.

Crowd-sourcing interface. To ensure high quality responses, we carefully design a crowd-sourcing protocol, while following common quality control mechanisms [10, 28, 39, 40, 45, 49, 56, 66]. First, we provide a set of in-

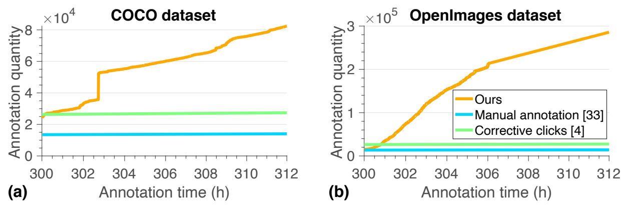

Figure 9. **Experiments on (a) COCO and (b) OpenImages.** Annotation quantity vs. annotation time for [33], [4] and ours.

structions with examples; then, the annotators must pass a qualification test to proceed to the annotation stage; and finally, we monitor performance by using hidden quality control images. The annotators are shown a mask and a class label. They should respond positively if the mask outlines the object correctly (IoU ≥ 0.75), and negatively otherwise.

Data collection. We train our segmentation model on 1,000 fully annotated images from the ADE20K training set [68] and obtain more than 10M mask predictions on the 1M unlabeled Places. Then, we run our framework using all design choices and hyper-parameter values from Sec. 4-5. We run in total 191,929 verification questions on Amazon Mechanical Turk, resulting in 730 high quality clusters ($\tilde{Q} \geq 0.85$). We accept in total 993,677 high quality object segmentation masks, which form our annotated dataset (Fig. 10). The set of our annotations is 101$\times$ larger than our initial training set and 2.3$\times$ larger than ADE20K [68].

Annotation time. The mean response time of the annotators is 1.4 s per binary question. The total time of the verification including quality control is 73 hours. Adding this time to the overhead for segmenting the initial set results in 290 annotation hours or less than \$3,000. Note that manually drawing 1M polygons would require 22k hours and cost over \$200k [33], while the interactive segmentation of OpenImages [4] would require about 11k hours. Our framework leads to a 76$\times$ speed up in time compared to [33].

Quality. To evaluate the segmentation quality SQ of our masks, we randomly select 1,142 images from the unlabeled Places pool and an expert annotator manually draws accurate polygons around each instance. We achieve an SQ of 81.4% computed with 1,109 matching GT annotations. The quality of our masks is on par with that of other datasets: 82.6% in COCO [23], 83.9% in Cityscapes [24], 77.9% in Vistas [24], 84.0% in OpenImages [4], 85.0% in LVIS [13] and 85.9% in ADE20K [24]. Note that we use these numbers as a reference and not for straight comparison because the size of the instances, the number of classes and the scene complexity vary a lot across datasets. In Sec. 5.3, we present a more fair comparison to [33] and [4].

Annotation usefulness. We evaluate the practical value of the obtained annotations by training PointRend [25] on 1M masks (Places masks and about 10k initial masks). Training a vanilla model from sparsely annotated images can be

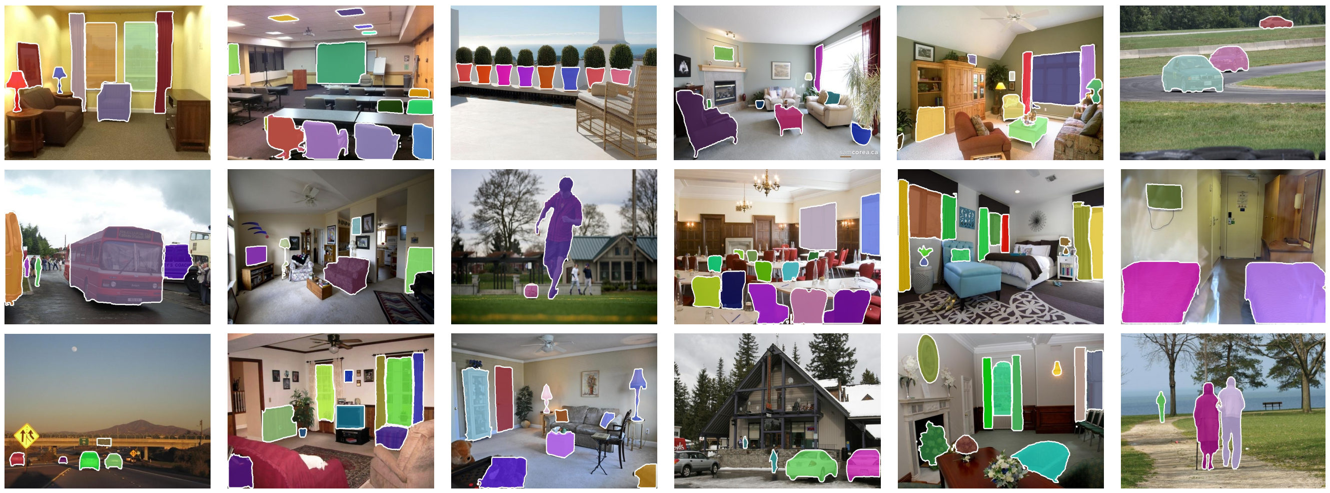

Figure 10. **Obtained mask annotations in Places** using our proposed interactive approach under a small fixed annotation budget.

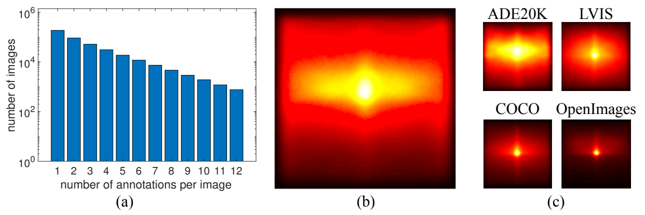

Figure 11. **Statistics on Places annotations.** (a) **Density:** The number of our annotations per image. **Diversity:** The distribution of (b) our annotation centers compared to (c) ADE20K, LVIS, COCO and OpenImages V6 train sets.

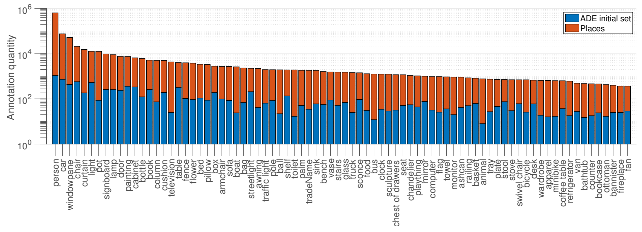

Figure 12. **Object class distribution.** The number of annotations for the initial set (blue) and the obtained Places annotations (red).

catastrophic as proposals are sampled from unlabeled image regions and used as negatives for the classifier [38, 57, 63]. We do two simple modifications to avoid this. First, we lower the proposals batch size so that the fraction of positives and negatives is the same as in a fully annotated image. Second, we use the predicted non-accepted masks to avoid sampling negative proposals that highly overlap with them. We achieve 12.7% AP on the ADE20K validation set. As a reference, training on the initial fully annotated set achieves 8.1% AP. Therefore, at a modest extra annotation time of 73 h we obtain a much better model. From another perspective, spending 73 h to manually annotate more objects and train a model on 1,400 manually annotated images achieves 9.6% AP. This shows that given the same annotation time, our method leads to a better model than manual annotation.

Annotation coverage. We obtain an annotation coverage (fraction of instances found) of 9.7% in Places (estimated on the annotated 1,142 images). A higher budget could lead to more masks since we did not fully annotate the trees. As a reference, we obtain a coverage of 27.1% in ADE20K.

Density and diversity. Fig. 11(a) shows the histogram with the number of annotations per image. Fig. 11(b) shows the spatial distribution of our obtained mask centers in normalized image coordinates. Even though none of our masks were drawn manually, we observe a quite diverse spatial distribution of complexity. Fig. 12 shows the object class distribution of our annotations compared to the initial training set. With an extra time of only 73h, we obtain two orders of magnitude more annotations for most of the classes.

7. Conclusions

We presented a highly efficient pipeline for annotating object segmentation masks. Our pipeline overpasses the major limitation of manual or interactive annotation where the cost grows linearly with the number of annotations. Instead, it exploits the commonalities between objects at a large scale and quickly propagates few verification labels to many examples. By applying our pipeline to a large-scale experiment, we obtained 1M masks with only 290 annotation hours. Our work marks a new direction for exploring the scaling of instance annotation. Future work involves turning the obtained sparsely labeled images into fully-annotated ones and annotating uncommon classes appearing in the long tail distribution as well as stuff classes.

Acknowledgments. This work is supported by the Mitsubishi Electric Research Laboratories. We thank V. Kalogeiton for helpful discussion and proofreading.

References

[1] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018. 1, 2

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

[3] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. What's the point: Semantic segmentation with point supervision. In *ECCV*, 2016. 2

[4] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019. 1, 2, 4, 5, 6, 7

[5] Y. Boykov and M. P. Jolly. Interactive graph cuts for optimal boundary and region segmentation of objects in N-D images. In *ICCV*, 2001. 2

[6] Yuri Boykov and Vladimir Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. on PAMI*, 26(9):1124–1137, 2004. 2

[7] L. Castrejon, K. Kundu, R. Urtasun, and S. Fidler. Annotating object instances with a Polygon-RNN. In *CVPR*, 2017. 1, 2

[8] M. Cordts, M. Omran, S. Ramos, T. Rehfeld, M. Enzweiler, R. Benenson, U. Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016. 1, 2

[9] J. Dai, K. He, and J. Sun. Boxsup: Exploiting bounding boxes to supervise convolutional networks for semantic segmentation. In *ICCV*, 2015. 2

[10] Ian Endres, Ali Farhadi, Derek Hoiem, and David A Forsyth. The benefits and challenges of collecting richer object annotations. In *DeepVision workshop at CVPR*, 2010. 7

[11] M. Everingham, S. Eslami, L. van Gool, C. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge: A retrospective. *IJCV*, 2015. 2

[12] M. Guillaumin, D. Küttel, and V Ferrari. ImageNet auto-annotation with segmentation propagation. *IJCV*, 2014. 2

[13] Agrim Gupta, Piotr Dollar, and Ross Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1, 2, 5, 7

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 4

[15] Q. Huang, M. Han, B. Wu, and S. Ioffe. A hierarchical conditional random field model for labeling and segmenting images of street scenes. In *CVPR*, 2011. 2

[16] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019. 3

[17] S. Jain and K. Grauman. Click carving: Segmenting objects in video with point clicks. In *Proceedings of the Fourth AAAI Conference on Human Computation and Crowdsourcing*, 2016. 2

[18] Suyog Dutt Jain and Kristen Grauman. Active image segmentation propagation. In *CVPR*, 2016. 2

[19] Ajay J Joshi, Fatih Porikli, and Nikolaos Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009. 2

[20] Ashish Kapoor, Kristen Grauman, Raquel Urtasun, and Trevor Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007. 2

[21] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1

[22] Li Yi1 Vladimir G Kim, Duygu Ceylan, I-Chao Shen3 Mengyan Yan, Hao Su1 Cewu Lu1 Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. In *SIGGRAPH*, 2016. 2

[23] A. Kirillov. panoptic segmentation dataset overview. 2018. 7

[24] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 5, 7

[25] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 3, 4, 7

[26] Adriana Kovashka, Sudheendra Vijayanarasimhan, and Kristen Grauman. Actively selecting annotations among objects and attributes. In *ICCV*, 2011. 2

[27] D. Kuettel, M. Guillaumin, and V. Ferrari. Segmentation Propagation in ImageNet. In *ECCV*, 2012. 2

[28] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Tom Duerig, et al. The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv preprint arXiv:1811.00982*, 2018. 3, 7

[29] Victor Lempitsky, Pushmeet Kohli, Carsten Rother, and Toby Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009. 2

[30] Zhuwen Li, Qifeng Chen, and Vladlen Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018. 1, 2, 3

[31] D. Lin, J. Dai, J. Jia, K. He, and J. Sun. Scribblesup: Scribble-supervised convolutional networks for semantic segmentation. In *CVPR*, 2016. 2

[32] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 4

[33] 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 *ECCV*, 2014. 1, 2, 4, 6, 7

[34] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019. 1, 2

[35] Oisin Mac Aodha, Neill DF Campbell, Jan Kautz, and Gabriel J Brostow. Hierarchical subquery evaluation for active learning on a graph. In *CVPR*, 2014. 2

[36] Kevis-Kokitsi Maninis, Sergi Caelles, Jordi Pont-Tuset, and Luc Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 2

[37] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 2

[38] Yusuke Niitani, Takuya Akiba, Tommi Kerola, Toru Ogawa, Shotaro Sano, and Shuji Suzuki. Sampling techniques for large-scale object detection from sparsely annotated objects. In *CVPR*, 2019. 8

[39] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 1, 2, 7

[40] Dim P Papadopoulos, Jasper RR Uijlings, Frank Keller, and Vittorio Ferrari. Training object class detectors with click supervision. In *CVPR*, 2017. 2, 7

[41] Dim P. Papadopoulos, Jasper R. R. Uijlings, Frank Keller, and Vittorio Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*, 2016. 3, 6, 7

[42] 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. 2

[43] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004. 1, 2

[44] Michael Rubinstein, Ce Liu, and William T Freeman. Annotation propagation in large image databases via dense image correspondence. In *ECCV*, 2012. 2

[45] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. Berg, and L. Fei-Fei. ImageNet large scale visual recognition challenge. *IJCV*, 2015. 1, 7

[46] B. C. Russell, K. P. Murphy, and W. T. Freeman. LabelMe: a database and web-based tool for image annotation. *IJCV*, 2008. 1

[47] Behjat Siddiquie and Abhinav Gupta. Beyond active noun tagging: Modeling contextual interactions for multi-class active learning. In *CVPR*, 2010. 2

[48] Samarth Sinha, Sayna Ebrahimi, and Trevor Darrell. Variational adversarial active learning. In *ICCV*, 2019. 2

[49] Alexander Sorokin and David Forsyth. Utility data annotation with amazon mechanical turk. In *Workshop at CVPR*, 2008. 7

[50] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 1

[51] Yuandong Tian, Wei Liu, Rong Xiao, Fang Wen, and Xiaoou Tang. A face annotation framework with partial clustering and interactive labeling. In *CVPR*, 2007. 2

[52] A. Torralba and A. Efros. An unbiased look on dataset bias. In *CVPR*, 2011. 1

[53] Sudheendra Vijayanarasimhan and Kristen Grauman. Multi-level active prediction of useful image annotations for recognition. In *NIPS*, 2008. 2

[54] Sudheendra Vijayanarasimhan and Kristen Grauman. What's it going to cost you?: Predicting effort vs. informativeness for multi-label image annotations. In *CVPR*, 2009. 2

[55] Sudheendra Vijayanarasimhan and Kristen Grauman. Large-scale live active learning: Training object detectors with crawled data and crowds. *IJCV*, 108(1-2):97–114, 2014. 2

[56] Carl Vondrick, Donald Patterson, and Deva Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2013. 7

[57] Tiancai Wang, Tong Yang, Jiale Cao, and Xiangyu Zhang. Co-mining: Self-supervised learning for sparsely annotated object detection. *arXiv*, 2020. 8

[58] Maggie Wigness, Bruce A Draper, and J Ross Beveridge. Efficient label collection for unlabeled image datasets. In *CVPR*, 2015. 2

[59] Jiajun Wu, Yibiao Zhao, Jun-Yan Zhu, Siwei Luo, and Zhuowen Tu. Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation. In *CVPR*, 2014. 2

[60] Ning Xu, Brian Price, Scott Cohen, Jimei Yang, and Thomas S Huang. Deep interactive object selection. In *CVPR*, 2016. 2

[61] Angela Yao, Juergen Gall, Christian Leistner, and Luc Van Gool. Interactive object detection. In *CVPR*, 2012. 2

[62] Donggeun Yoo and In So Kweon. Learning loss for active learning. In *CVPR*, 2019. 2

[63] Jihun Yoon, Seungbum Hong, Sanha Jeong, and Min-Kook Choi. Semi-supervised object detection with sparsely annotated dataset. *arXiv*, 2020. 8

[64] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 2

[65] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 2, 6, 7

[66] B. Zhou, A. Lapedriza, A. Khosla, A. Oliva, and A. Torralba. Places: A 10 million image database for scene recognition. *IEEE Trans. on PAMI*, 2017. 7

[67] Bolei Zhou, Agata Lapedriza, Jianxiong Xiao, Antonio Torralba, and Aude Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 1, 2, 7

[68] B. Zhou, H. Zhao, X. Puig, S. Fidler, A. Barriuso, and A. Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 1, 2, 4, 5, 7

[69] Xiangxin Zhu, Carl Vondrick, Charless C Fowlkes, and Deva Ramanan. Do we need more training data? *IJCV*, 2016. 1