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 human-in-the-loop scheme that efficiently searches through the hierarchy of clusters and actively 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 the Places dataset with object segmentation masks for 100 object classes, we show that (1) given a budget of only $1,000, our scheme produces 4.3M object segmentation masks; (2) we reduce the annotation time by 830× compared to manual annotation; (3) the quality of our masks is on par with those from available manually annotated datasets.

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 [43], Places [63] 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 [48, 50, 65].

Manually annotating images for instance segmentation is much more expensive than image-level labeling as it requires humans to draw a detailed outline around every object in an image [8, 13, 32, 44, 64]. For example, annotating COCO [32] required 80s per segmentation mask, a single image in Cityscapes [8] took 1.5 hours, and the annotation of ADE20K [64] by a single annotator took several years. At this pace, constructing a dataset with 10M object masks would require more than 200k hours and cost over $2M [32].

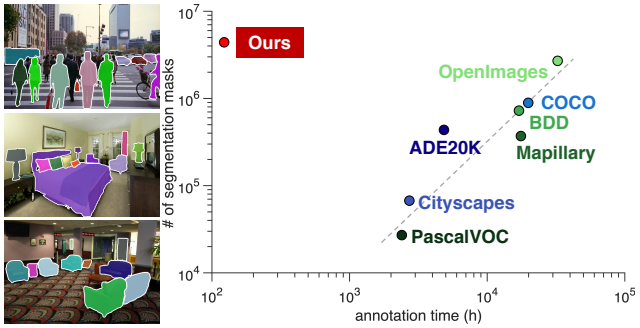

Figure 1. **Scaling up instance annotation.** (Left) Obtained object mask annotations on the Places images using our proposed framework. (Right) The size vs. the total annotation time in hours of our annotations compared to the most popular instance segmentation dataset. We reduce annotation cost by two orders of magnitude. Note that both axes are in log scale.

An alternative to manual annotation is interactive segmentation [1, 4, 7, 29, 33, 37, 41], where the human interaction is much faster (e.g., bounding boxes, clicks, scribbles, interactive polygons). Given this faster interaction, these methods aim to automatically infer the final object mask. They offer substantial gains in annotation time and can lead to larger datasets (OpenImages V5 [4]). However, because annotators intervene on every instance, the cost grows linearly with the number of annotations and therefore, at a larger scale, they become unaffordable.

In this paper, we propose a framework for crowdsourcing object segmentation masks that reduces the annotation time by 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 based on their similarity. Human annotators only need to *verify* a few

*Indicates equal contribution

masks per cluster and we then *propagate* the verification labels to the whole cluster. Since annotators interact with few instances per cluster, our cost scales with the number of clusters rather than masks. Annotation cost and annotation time are proportional, so in this paper we will use annotation time as it is a more objective measure than cost.

Given a small set of images with ground-truth mask annotations and a small annotation budget, our goal is to populate a large set of unlabeled images with high-quality segmentation masks (Sec. 4). We first train an instance segmentation model and obtain mask predictions on the unlabeled set. We then cluster class-specific masks based on their similarity using hierarchical clustering. Starting from the root of the tree, we search for candidate clusters likely to contain high-quality masks. Human annotators are asked quick binary verification questions for 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 annotated pool to retrain the model and the above steps are repeated iteratively (Fig. 2).

We first conduct simulated experiments to explore the design space of our framework (Sec. 5). Then, we conduct a large-scale experiment to populate the images of the Places dataset [63] with segmentation masks for 100 object classes (Sec. 6). Our results show that (1) we obtain 4.3M masks with a budget of less than $1,000 forming the largest annotation set for instance segmentation. (Note that a higher budget could lead to more masks.); (2) we reduce the annotation time by 830× compared to manual annotation [32]; (3) we show that the quality of our masks is on par with those from manually annotated datasets. We will release the framework code and the annotations upon acceptance.

2. Related Work

Instance segmentation datasets are built by manually drawing accurate polygons around object instances [8, 11, 32, 36, 60, 64]. The most popular examples are PASCAL VOC [11], ADE20K [64], COCO [32], LVIS [13], Cityscapes [8], Mapilary Vistas [36] and BDD100K [60]. An exception is OpenImages V5 [4], where the masks are collected using an interactive segmentation approach. Fig. 1 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 a factor of 10 would require 200k hours and $2M [32]). Our framework can produce segmentation masks 830× faster than [32] (Sec. 6).

Interactive object segmentation started in the early 2000's with the seminal work of graph cut and iterative graph cuts [5, 6, 41]. After that, many approaches were proposed to reduce the manual annotation effort using bounding boxes [9, 28, 56], point clicks [3, 4, 17, 29, 35, 38, 37, 57], scribbles [2, 3, 30], and editing polygons [7, 1, 33]. These methods can offer remarkable gains in time compared to the manual drawing of polygons. However, because they all require human intervention (e.g., click, scribble or editing) in every single object, the annotation cost still grows linearly with the number of object segmentation masks.

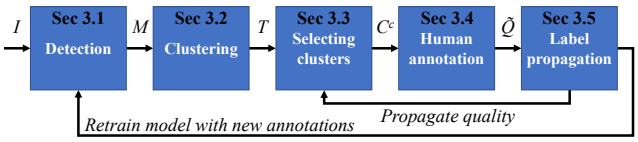

Figure 2. **Method overview.** Our proposed framework iteratively alternates between detecting object segmentation masks on an unlabeled set of images, clustering the masks into a tree, selecting clusters for annotation, asking humans to verify a few masks per cluster, and propagating verification quality to the whole cluster.

Annotation propagation uses pre-segmented images to guide the segmentation of new ones by propagating the annotation signal [42, 26, 12, 18, 40]. In [26], segmentation masks for the source classes are obtained by propagating masks from related classes, while in [18] an active approach selects which images should be annotated and which ones should be used for propagation.

Group-based labeling assigns a single labeled image to a whole group simultaneously [15, 49, 55]. LSUN [61] was built using human annotators that label only a few images, which are then used to train binary classifiers and automatically label many images at once with high confidence score. In Sec. 5.2 we modify [61] to work for instance segmentation and compare it with our framework.

Active learning has been used in computer vision for the tasks of image classification [19, 20, 25, 34, 46, 59], object detection [53, 58] and semantic segmentation [18, 45, 51, 52]. 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 fully 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 segmentation pipeline consists of five steps (Fig. 2): (a) Detection: we deploy an instance segmentation model on an unlabeled set to obtain a set of object segmentation masks (Sec. 4.1). (b) Clustering: Class-specific masks are hierarchically clustered using a feature representation to obtain a tree (Sec. 4.2). (c) Selecting clusters: We efficiently search the tree and select candidate clusters that are likely

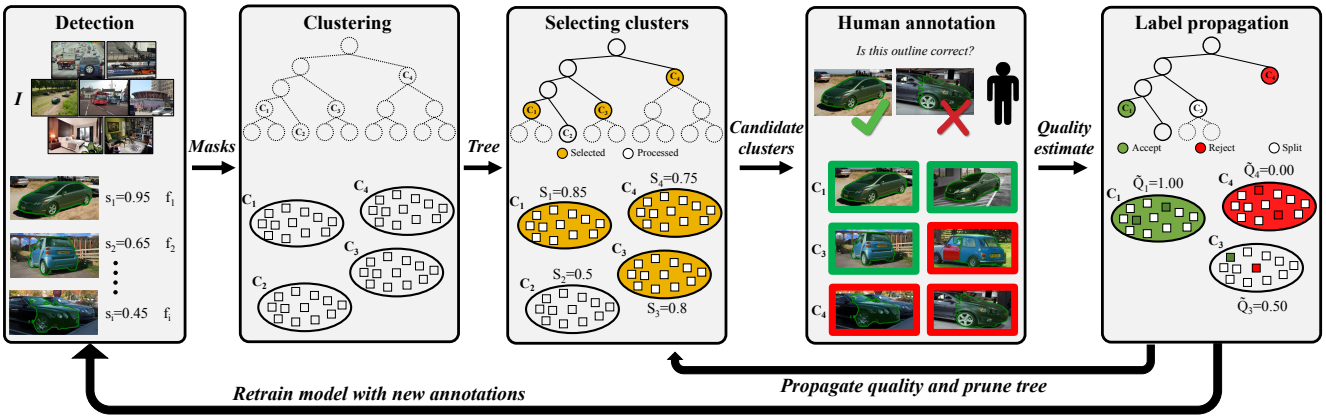

Figure 3. **The proposed human-in-the-loop framework.** We apply an instance segmentation model on an unlabeled set of images to detect object segmentation masks (***Detection***). Then, we use hierarchical clustering to cluster masks and form a tree of clusters (***Clustering***). We efficiently search 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***).

to contain high-quality masks (Sec. 4.3). (d) Human annotation: For each candidate cluster, we sample a few masks and ask human annotators to quickly verify whether they are correct or not. The annotators are instructed to respond positively if the mask correctly outlines the target object. We use the verification labels to estimate the quality of the candidate clusters (Sec. 4.4). (e) Propagation: If the estimated quality is very high or very low (i.e, the cluster almost exclusively contains only correct or wrong masks), we propagate the verification labels to the whole cluster and set this cluster as a leaf. Otherwise, we further split the cluster into its children. (Sec. 4.5). Our framework iteratively repeats steps (c), (d) and (e) to discover high-quality clusters with as few questions as possible. We add masks from high-quality clusters to our obtained dataset, retrain the instance segmentation model, and repeat the framework iteratively.

4. Method

In this section, we present our human-in-the-loop framework to quickly and efficiently populate unlabeled images with object segmentation masks.

4.1. Detection

In this step, we deploy an instance segmentation model to obtain object segmentation masks M on the unlabeled set of images I. For the instance segmentation model, we use PointRend [24]. 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 [29]. We multiply this IoU score with the classification confidence score of the box head to obtain a predicted score s for every predicted mask m. Therefore, for every predicted mask m, the output of the instance segmentation model has a score s and a deep feature representation f. f and s together can encode both the visual appearance and the mask quality. In Sec. 5.1, we evaluate different choices for a feature representation $\mathcal{F}$ to cluster with.

At the first iteration, the model is trained on a small set of fully annotated images with ground-truth masks. At each following step, the model is retrained using the initial annotated set together with all new annotation obtained from the unlabeled pool.

4.2. Clustering

In this step, we cluster M using a feature representation $\mathcal{F}$. 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. Hierarchical clustering forms $r = |C| = 2|M| - 1$ clusters. 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 $j \in [1, |M_j|]$. 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 human annotators to interactively find the final set of clusters.

4.3. Selecting clusters

In this step, we search T starting from the root node C_r to discover a set of high-quality clusters C^a that are accepted and added to the dataset. We use a priority queue to guide the search algorithm towards clusters that are more likely to be of high quality. Different search strategies lead to a different orderings of the clusters, which results in a trade-off between cost and number of masks obtained during searching. However, they all eventually lead to the same

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
  Pop C_j from C^c
  Check S_j to early reject or split
  Estimate quality $\tilde{Q}_j$ of C_j `// annotate`
  if $\tilde{Q}_j \geq K_a$ **then**
    `// accept`
    $C^a = C^a \cup C_j$
  else if $\tilde{Q}_j \leq 1 - K_a$ **then**
    `// reject`
  else
    `// split`
    Children $\{C_{left}, C_{right}\} = C_j$
    $C^c = C^c \cup \{C_{left}, C_{right}\}$
  end
end
return C^a

trade-off because they visit the same clusters at the end of the procedure.

The majority of the clusters in the tree are not of high quality and blindly annotating every cluster will lead to an inferior trade-off between cost and number of annotations. For this reason, we actively select which clusters to annotate. To do this, we estimate the likelihood of a cluster to be of high quality given the scores of the masks it contains.

A mask m_i is considered correct when its intersection over union (IoU) with a perfect ground truth 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 likelihood probabilities P_h and P_l on a held-out set of images with ground-truths. We denote two thresholds K_{pr} and K_{pa} on S_j such that $P(Q_j \leq 1 - K_a | S_j < K_{pr}) \approx 1$ and $P(Q_j \geq K_a | S_j < K_{pa}) \approx 0$. When $S_j \leq K_{pr}$, the cluster is automatically rejected and when $K_{pr} < S_j \leq K_{pa}$, the cluster is split to its children. The candidate clusters C^c that are selected for further human verification are only those with $S_j > K_{pa}$ (Fig. 4). Alg. 1 outlines this procedure.

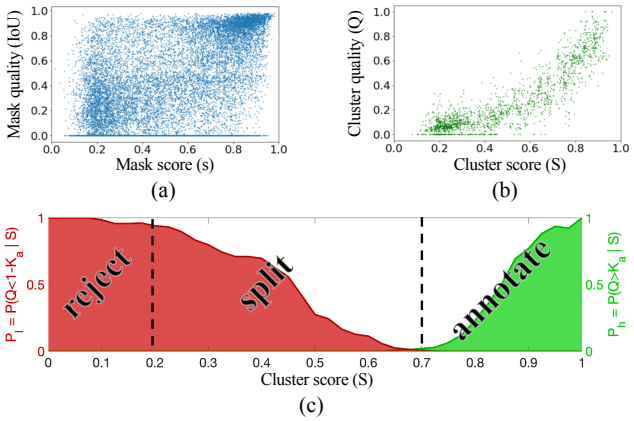

Figure 4. **Selecting cluster to annotate.** (a) The real IoU of the mask as a function its mask score s. (b) The real cluster quality (Q) as a function of the cluster score S. We observe a much stronger correlation at a cluster level. (c) The estimated likelihood probabilities P_h and P_l given S. We only select clusters to annotate when $P_h > 0$. When $P_l \approx 1$, the cluster is rejected, otherwise it is split without any human intervention.

4.4. Human annotation

In this step, we ask human annotations 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 seconds [27, 39]. Our key difference is that the annotator only verifies N_s randomly sampled masks from each candidate cluster 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 verification responses of the N_s sampled masks $\hat{M}$ to obtain an estimated quality $\tilde{Q}$ for a cluster as $\tilde{Q} = \frac{1}{N_s} \sum_{l \in \hat{M}} l$.

4.5. 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 $\tilde{Q}_j < 1 - K_a$, we *reject* the cluster and we propagate the negative verification labels ($l_i = 0$) to all the masks of C_j. 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 our 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 frame-

work (Sec. 4) and find optimal parameters to minimize the annotation cost given the desired annotation quality. In Sec. 5.1, we perform experiments on ADE20K [64], while in Sec. 5.2 we simulate a realistic large-scale scenario to estimate various parameters not possible to estimate within a small dataset and to compare our method with various alternatives at large scale.

Implementation details. For all experiments, we use PointRend [24] with Feature Pyramid Network [31] and ResNet50 [14] as our instance segmentation model. We obtain predictions after applying NMS and keep the ones with a score above 0.2 and bounding box area above 1000 pixels.

5.1. Simulated experiments on ADE

Dataset. We use the training set of ADE20K [64], which consists of 20,210 images and ground-truth masks for 100 object classes. We use half of this set to initially train the instance segmentation model (10,105 images) and we consider the other half as our unlabeled pool. The validation set (2,000 images) is a held-out set which we use to measure the performance of the model.

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 required 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 the annotated clusters.

For all experiments in this section, we factor out the sampling step of the annotation stage and we assume that when we annotate a cluster, $\tilde{Q} = Q$. We consider a mask with IoU ≥ 0.75 with the ground-truth as correct, and wrong otherwise and we set $K_a = 0.85$. The effect of these parameters and of the sampling are analyzed in Sec. 5.2. For simplicity, we also perform one iteration of our framework for most of the experiments and analyze the effect of retraining the model at the end of this section.

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 mask logits with the backbone features. We obtain different numbers of clusters from each T by cutting it 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 three feature types. Combining it with the mask score s (black line) leads to better clustering (more annotations given the same number of clusters); we use this in the remainder of the paper.

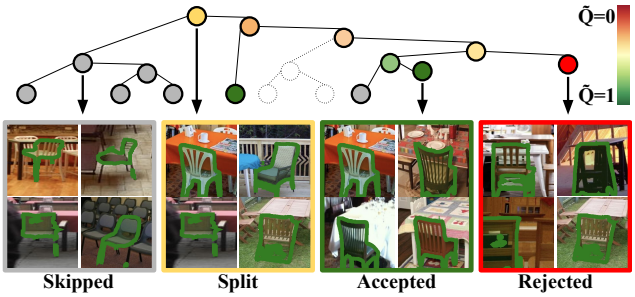

Figure 5. **Examples of clusters in tree.** We show clusters in Places for the chair class. Clusters are *skipped* with low cluster score S, *split* when containing both low and high quality masks, *accepted* when mostly correct, and *rejected* otherwise. The dotted clusters have yet to be explored and have no quality estimate $\tilde{Q}$.

Searching the tree. We evaluate the *universal* thresholding under different number of clusters and four main search algorithms to explore and interactively 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 over children clusters that are more likely to be of high quality; (4) *Heuristics only*: always sort all candidate clusters according to the heuristic score in decreasing order while searching the tree. For a fair comparison, we factor out the active cluster selection and we select all the clusters for annotation.

In Fig. 6(b), we observe that most of the searching algorithms achieve a much better trade-off than the universal thresholding (blue line). All searching algorithms reach the same accuracy at the end of the curves because the only difference is the search ordering and the searching always leads to the same leaves. We observe that *Heuristics only: Mask score* achieve 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 that 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. All experiments above assume that all clusters that are explored during searching are also selected for verification. We evaluate here the active selection of these clusters by following our active selection mechanism described in Sec. 4.3 using different values for K_{pa}. In Fig. 6, we observe that very high values of K_{pa} result in not selecting high-quality clusters for annotations leading to a very low number of good annotations (orange line),

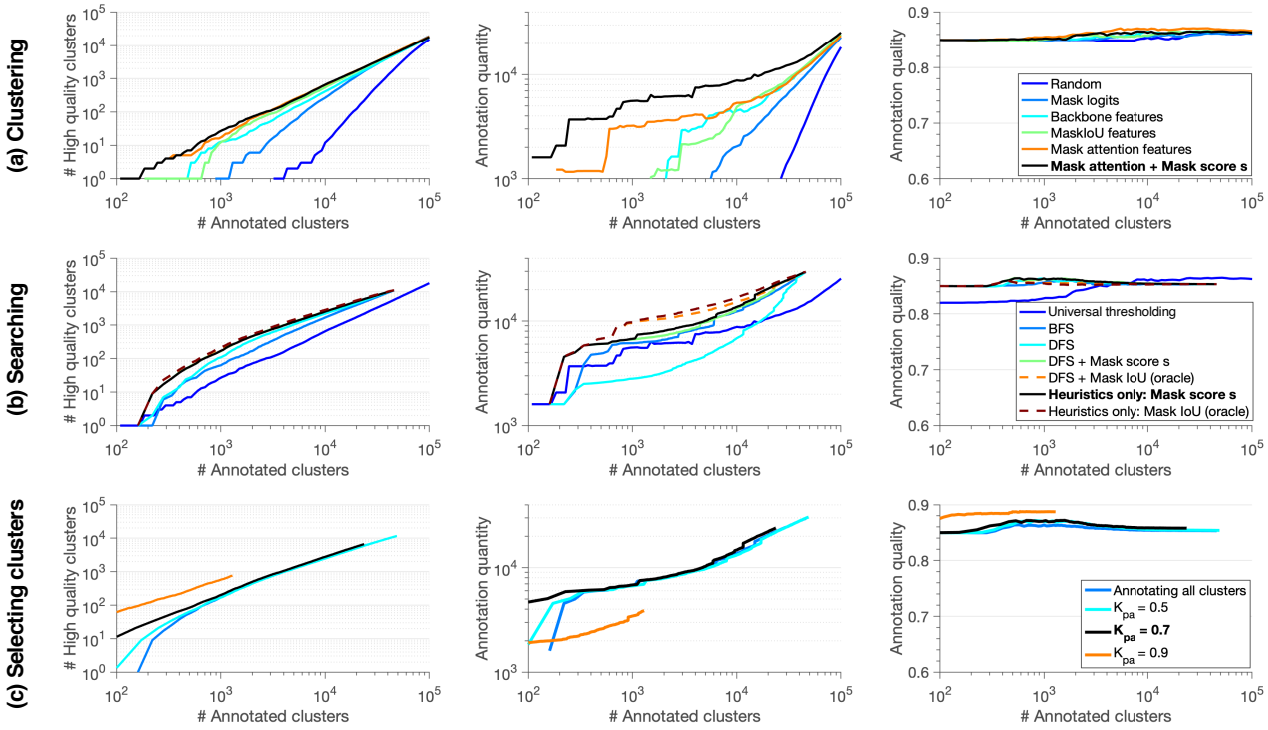

Figure 6. **Experimental results on ADE20K.** We show the number of high-quality clusters, the annotation quantity and the annotation quality versus the number of annotated clusters in log-log scale. **(a) Clustering**: The effect of using different feature representations $\mathcal{F}$ to construct 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.

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).

Iterative framework. We evaluate here the effect of retraining the instance segmentation model and performing our pipeline in a loop. It has been shown that the training data need to grow significantly for an accuracy improvement in the training models [48, 50, 65]. For this reason, we use a different setting here where only 1,000 ADE training images are used to train the instance segmentation model and we consider the other 19,210 images as our unlabeled pool. In Fig. 7, we report the annotation quantity in the unlabeled pool and the AP performance of the segmentation model in the validation set over the iterations. We observe that retraining the model iteratively leads to gradually increasing the performance of the instance segmentation model (better AP). This shows the usefulness of the obtained annotations. The model predicts new candidate masks at every iteration and our framework can find new high quality mask that are added to the labeled dataset. Note that between rounds we filter highly overlapping detections to maintain at most one annotation per object.

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 that directly affect both the quality and the cost, namely the number of verified samples per cluster N_s, the cluster quality threshold K_a, and the mask IoU threshold K_{iou}. These parameters cannot be optimized directly in ADE20K due to its small size.

Evaluation set. The number of ground-truth masks per class in ADE20K varies from a few thousands 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 the detection masks at a larger scale follow a similar joint distribution $P(f, s, IoU(m))$. We learn this distribution on ADE20K in a class-specific way using a Gaussian Mixture model. Then, we sample triplets $\{f_i, s_i, IoU(m_i)\}$ from these distributions and we form the simulated class-specific sets by following the class distribution of ADE20K. For the sizes of the sets, we use the ratio between the number of images in Places [63] over the number of images in ADE20K. This leads in $100M$ $\{f_i, s_i, IoU(m_i)\}$ triplets that we use to run our proposed framework in simulation.

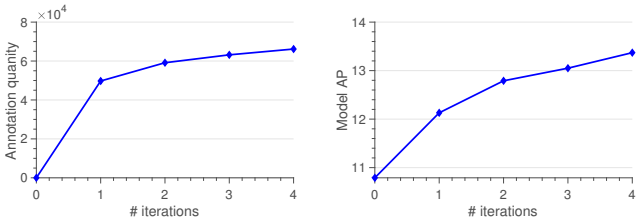

Figure 7. **Retraining model with new annotations.** (Left) Our framework leads to higher annotation quantity when we run it iteratively. (Right) Re-training the instance segmentation model with more obtained annotations leads to a better AP.

Annotation and propagation. Following the human annotation consistency in instance segmentation datasets [4, 13, 23, 64], a high quality dataset typically has a segmentation quality SQ between 0.8 and 0.85. SQ is defined as the average IoU over 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 estimate of the cluster quality $\tilde{Q}$ 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 low number of annotations. We run our pipeline for different values of these parameters and we aim at keeping $SQ \geq 0.85$, while first minimizing N_s and then from all possible solutions for the pair K_a and K_{iou} we keep the one that results in the largest number of obtained annotations. This results in N_s=15, K_a=0.85, and K_{iou}=0.75.

Comparison to other annotation approaches. We compare the performance of our pipeline to other alternatives for annotation using the large-scale simulation. In Fig. 8, we report the trade-off between annotation quantity and quality and the annotation cost in hours assuming 2 s for each binary question and 80 s for each manual drawing [32]. Our framework leads to a speed up of two orders of magnitude comparing to [32], while maintaining a high quality. We modified the method used for LSUN for instance segmentation [61]. We observe that our method achieves a better trade-off than [61], while maintaining a much better annotation quality. Even in case of the object class bed, where our pipeline achieves the smallest improvement over a naive brute-force binary verification [39], [61] leads to slightly more annotations but with a much lower quality.

6. Large scale experiments on Places

In this section, we present our results on a large-scale experiment to obtain object segmentation masks using our framework with a budget of less than $1,000.

Data. We use the trainval set of the Places dataset [63] as our unlabeled pool of images, which consists of 8,063,128 images and we populate them with high-quality masks for

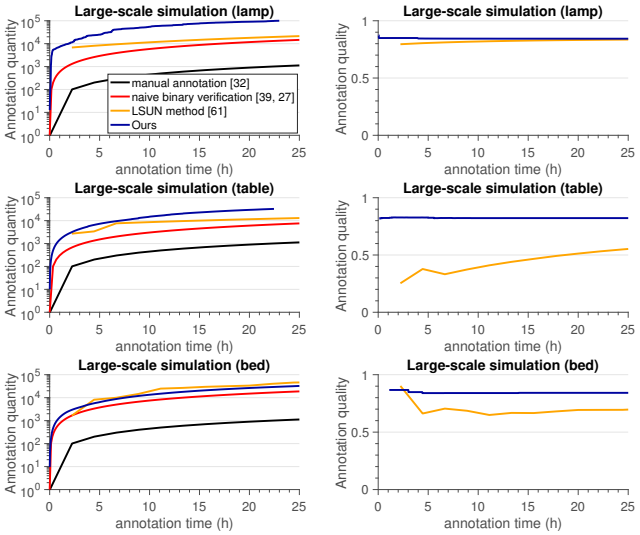

Figure 8. **Large-scale simulated experiment.** The annotation quantity and quality vs. the annotation time in hours for 3 object classes. We compare our pipeline with manual annotation [32], naive binary verification [39] and the LSUN method [61].

the 100 object classes of the ADE20K challenge [64].

Crowd-sourcing interface. To ensure high quality responses, we carefully design a crowd-sourcing protocol for the verification task, while following common quality control mechanisms [10, 27, 37, 38, 43, 47, 54, 62]. First, we provide a set of instructions with examples; secondly, 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 the train set of ADE20K and obtain more than 100M mask predictions on Places. Then, we run our framework by performing one iteration using all design and hyperparameters from Sec. 4-5. We run in total 245,177 verification questions on Amazon Mechanical Turk, resulting in 1,621 high quality clusters ($\tilde{Q} \geq 0.85$). These clusters contain 4,268,921 object segmentation masks, which form our annotated dataset (Fig. 9). The set of our obtained annotations is 9.8× larger than ADE20K [64] and 1.6× larger than OpenImages [4], the largest instance segmentation dataset.

Annotation time and cost. The mean response time of the annotators is 1.5 s per verification question. The total time including quality control is 114 hours requiring only $980. Note that manually drawing polygons for 4.3 M objects would require 95k hours and cost over $900k [32], while the interactive segmentation of [4] would require about 48k hours. Our framework leads to a 830× speed

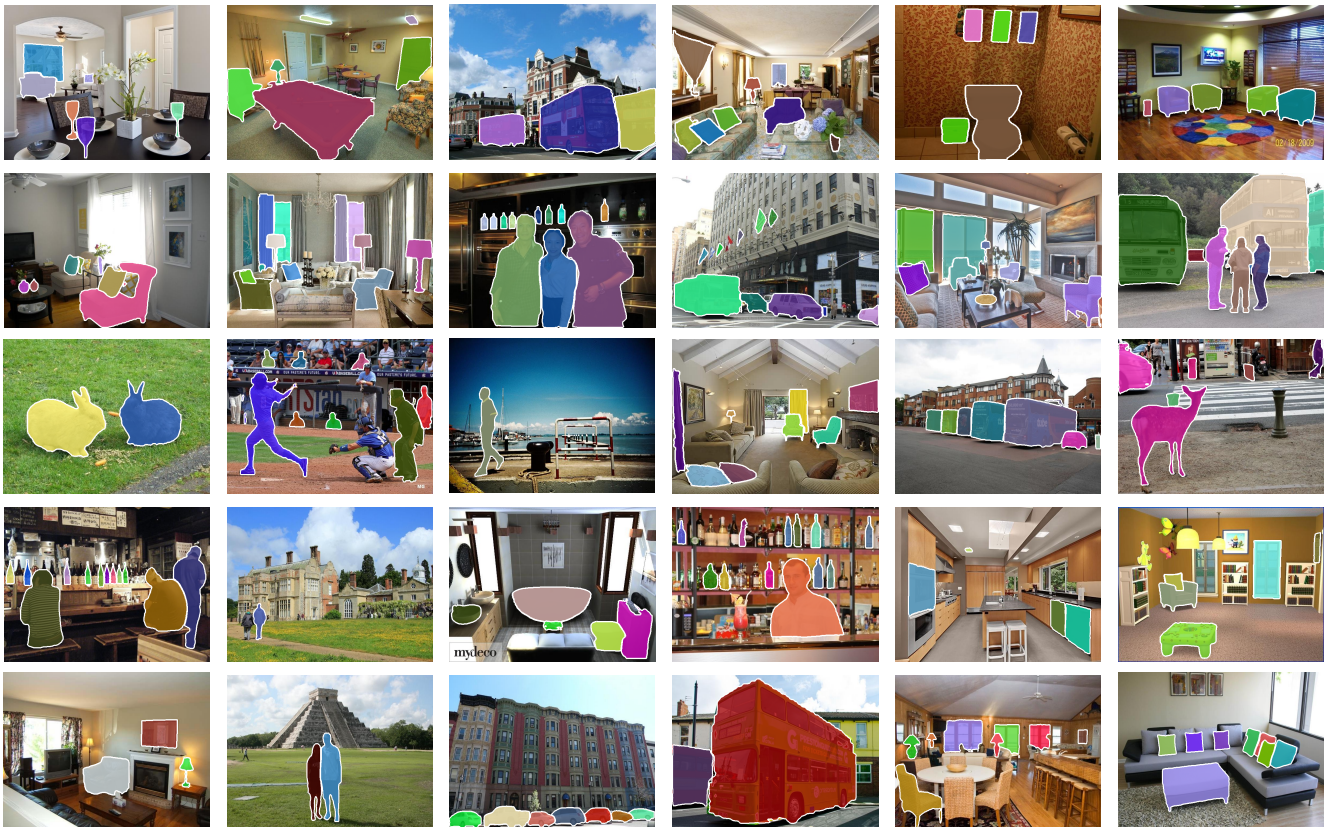

Figure 9. **Obtained mask annotations in Places.** We show here example images from the Places dataset with our obtained mask annotations using our pipeline under a small fixed annotation budget.

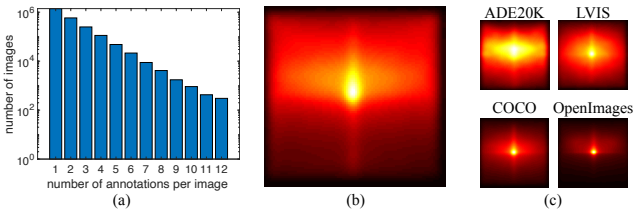

Figure 10. **Statistics on Places annotations.** (a) **Density:** The number of our annotations per image. **Diversity:** The distribution of (b) our annotation centers compared to (c) ADE20K, LVIS, COCO and OpenImages V6 train sets.

up in annotation time compared to manual annotation [32] and to a 425× speed up compared to OpenImages [4].

Quality. To evaluate the segmentation quality SQ of our masks, we randomly select 1,142 images from Places and an expert annotator manually draws accurate polygons around each instance. We achieve an SQ of 82.3% computed with 891 matching GT annotations. The quality of our masks is on par with that of other datasets: 82.6% in COCO [22], 83.9% in Cityscapes [23], 77.9% in Vistas [23], 84.0% in OpenImages [4], 85.0% in LVIS [13] and 85.9% in ADE20K [23]. Note that we use these numbers as a reference and not for straight comparison because the number and the size of the instances, the number of classes and the scene complexity vary a lot across datasets.

Density and diversity. Fig. 10(a) shows the histogram with the number of annotations per image. Note that a higher annotation budget could lead to more masks and more densely annotated images. Fig. 10(b) shows he 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, greater than COCO and OpenImages.

7. Conclusion

We presented a highly efficient framework for annotating object segmentation masks. Our framework overpasses the major limitation of manual or interactive annotations 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 framework to a large-scale experiment, we obtained 4.3M annotation masks with a budget of less than $1,000. Our obtained annotations and framework code will be publically available upon acceptance. Our work marks a new direction for exploring the

scaling of instance annotation. Future work involves turning the obtained partially labeled images into fully-annotated ones, annotate uncommon object classes appearing in the long tail distribution and stuff classes.

References

[1] D. Acuna, H. Ling, A. Kar, and S. Fidler. Efficient interactive annotation of segmentation datasets with polygon-rnn++. In *CVPR*, 2018. 1, 2

[2] E. Agustsson, J. R. Uijlings, and V. 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] R. Benenson, S. Popov, and V. Ferrari. Large-scale interactive object segmentation with human annotators. In *CVPR*, 2019. 1, 2, 7, 8

[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] Y. Boykov and V. 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] I. Endres, A. Farhadi, D. Hoiem, and D. 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] A. Gupta, P. Dollar, and R. Girshick. Lvis: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1, 2, 7, 8

[14] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5

[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] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. 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] S. D. Jain and K. Grauman. Active image segmentation propagation. In *CVPR*, 2016. 2

[19] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *CVPR*, 2009. 2

[20] A. Kapoor, K. Grauman, R. Urtasun, and T. Darrell. Active learning with gaussian processes for object categorization. In *ICCV*, 2007. 2

[21] W. Kay, J. Carreira, K. Simonyan, B. Zhang, C. Hillier, S. Vijayanarasimhan, F. Viola, T. Green, T. Back, P. Natsev, et al. The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950*, 2017. 1

[22] A. Kirillov. panoptic segmentation dataset overview. 2018. 8

[23] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic segmentation. In *CVPR*, 2019. 7, 8

[24] A. Kirillov, Y. Wu, K. He, and R. Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 3, 5

[25] A. Kovashka, S. Vijayanarasimhan, and K. Grauman. Actively selecting annotations among objects and attributes. In *ICCV*, 2011. 2

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

[27] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, T. 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. 4, 7

[28] V. Lempitsky, P. Kohli, C. Rother, and T. Sharp. Image segmentation with a bounding box prior. In *ICCV*, 2009. 2

[29] Z. Li, Q. Chen, and V. Koltun. Interactive image segmentation with latent diversity. In *CVPR*, 2018. 1, 2, 3

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

[31] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 5

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 1, 2, 7, 8

[33] H. Ling, J. Gao, A. Kar, W. Chen, and S. Fidler. Fast interactive object annotation with curve-gcn. In *CVPR*, 2019. 1, 2

[34] O. Mac Aodha, N. D. Campbell, J. Kautz, and G. J. Brostow. Hierarchical subquery evaluation for active learning on a graph. In *CVPR*, 2014. 2

[35] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool. Deep extreme cut: From extreme points to object segmentation. In *CVPR*, 2018. 2

[36] G. Neuhold, T. Ollmann, S. Rota Bulo, and P. Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In *ICCV*, 2017. 2

[37] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari. Extreme clicking for efficient object annotation. In *ICCV*, 2017. 1, 2, 7

[38] D. P. Papadopoulos, J. R. Uijlings, F. Keller, and V. Ferrari. Training object class detectors with click supervision. In *CVPR*, 2017. 2, 7

[39] D. P. Papadopoulos, J. R. R. Uijlings, F. Keller, and V. Ferrari. We don't need no bounding-boxes: Training object class detectors using only human verification. In *CVPR*, 2016. 4, 7

[40] K. Rakelly, E. Shelhamer, T. Darrell, A. A. Efros, and S. Levine. Few-shot segmentation propagation with guided networks. *arXiv preprint arXiv:1806.07373*, 2018. 2

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

[42] M. Rubinstein, C. Liu, and W. T. Freeman. Annotation propagation in large image databases via dense image correspondence. In *ECCV*, 2012. 2

[43] 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

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

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

[46] S. Sinha, S. Ebrahimi, and T. Darrell. Variational adversarial active learning. In *ICCV*, 2019. 2

[47] A. Sorokin and D. Forsyth. Utility data annotation with amazon mechanical turk. In *Workshop at CVPR*, 2008. 7

[48] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *ICCV*, 2017. 1, 6

[49] Y. Tian, W. Liu, R. Xiao, F. Wen, and X. Tang. A face annotation framework with partial clustering and interactive labeling. In *CVPR*, 2007. 2

[50] A. Torralba and A. Efros. An unbiased look on dataset bias. In *CVPR*, 2011. 1, 6

[51] S. Vijayanarasimhan and K. Grauman. Multi-level active prediction of useful image annotations for recognition. In *NIPS*, 2008. 2

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

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

[54] C. Vondrick, D. Patterson, and D. Ramanan. Efficiently scaling up crowdsourced video annotation. *IJCV*, 2013. 7

[55] M. Wigness, B. A. Draper, and J. Ross Beveridge. Efficient label collection for unlabeled image datasets. In *CVPR*, 2015. 2

[56] J. Wu, Y. Zhao, J.-Y. Zhu, S. Luo, and Z. Tu. Milcut: A sweeping line multiple instance learning paradigm for interactive image segmentation. In *CVPR*, 2014. 2

[57] N. Xu, B. Price, S. Cohen, J. Yang, and T. S. Huang. Deep interactive object selection. In *CVPR*, 2016. 2

[58] A. Yao, J. Gall, C. Leistner, and L. Van Gool. Interactive object detection. In *CVPR*, 2012. 2

[59] D. Yoo and I. S. Kweon. Learning loss for active learning. In *CVPR*, 2019. 2

[60] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, 2020. 2

[61] F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 2, 7

[62] 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

[63] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In *NIPS*, 2014. 1, 2, 6, 7

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

[65] X. Zhu, C. Vondrick, C. C. Fowlkes, and D. Ramanan. Do we need more training data? *IJCV*, 2016. 1, 6