

# Clustering for Large-scale Segmentation Dataset Collection

by

Jeffrey Hu

Submitted to the Department of Electrical Engineering and Computer  
Science

in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science and Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2019

© Massachusetts Institute of Technology 2019. All rights reserved.

Author .....  
Department of Electrical Engineering and Computer Science  
August 23, 2019

Certified by .....  
Antonio Torralba  
Professor  
Thesis Supervisor

Accepted by .....  
Katrina LaCurts  
Chairman, Department Committee on Graduate Theses



# Clustering for Large-scale Segmentation Dataset Collection

by

Jeffrey Hu

Submitted to the Department of Electrical Engineering and Computer Science  
on August 23, 2019, in partial fulfillment of the  
requirements for the degree of  
Master of Science in Computer Science and Engineering

## Abstract

Segmentation datasets are smaller and much more expensive to collect than their image classification counterparts. Leveraging machine learning in the annotation process will be critical to scaling these datasets up.

In this thesis, we propose an iterative cluster-based approach to segmentation data collection. By using existing networks to predict millions of segmentations and clustering to group similar predictions together, we ask human annotators a small number of questions per cluster and collect a large number of reasonable-quality segmentations at low cost. Although the collected segmentations are biased towards objects already predicted by the network, we demonstrate that they improve performance upon re-training and that the procedure can be applied iteratively, up to a point, to discover harder and harder objects. We demonstrate this pipeline in simulation and show promising results on real unlabeled images. We also present a new annotation tool called LabelMeLite for the rapid filtering and editing of predicted segmentations.

Thesis Supervisor: Antonio Torralba  
Title: Professor



## Acknowledgments

I would like to thank Professor Antonio Torralba for the opportunities he has provided me with and for his patience and guidance of my research. Without him, this thesis would not have been possible.

I would also like to thank Dimitris Papadoupolos, Hang Zhao, and Xavier Puig for their contributions to this project. Without their support and advice, this project would also not have been possible.



# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
<b>2</b>	<b>Related Work</b>	<b>15</b>
2.1	Segmentation Datasets . . . . .	15
2.2	Smart Annotation Tools . . . . .	17
2.3	Semi-supervised Learning . . . . .	18
<b>3</b>	<b>Clustering for Segmentation Data Collection</b>	<b>19</b>
3.1	Overview . . . . .	19
3.2	Models . . . . .	20
3.2.1	MaskRCNN . . . . .	20
3.2.2	MaskScoringRCNN . . . . .	21
3.3	Simulation on ADE20K . . . . .	22
3.4	Clustering in Places . . . . .	24
3.5	Next Steps . . . . .	25
3.6	Future Work . . . . .	26
<b>4</b>	<b>LabelMeLite</b>	<b>27</b>
4.1	Motivation . . . . .	27
4.2	Main Interface . . . . .	28
4.3	Implementation . . . . .	29
4.3.1	COCO API . . . . .	29
4.4	Amazon Mechanical Turk . . . . .	30

4.4.1	Edit Interface . . . . .	30
4.4.2	Yes/No Interface . . . . .	31
4.5	Future Work . . . . .	31
<b>5</b>	<b>Conclusion</b>	<b>33</b>



# List of Figures

2-1	Plot of dataset size and annotation density for various segmentation datasets. Both metrics are desirable. . . . .	17
3-1	Visualization of the annotation pipeline. Clustering prevents the human annotator from being overwhelmed. . . . .	20
3-2	Model architecture for MaskRCNN. The features we use as our embedding are circled in red. . . . .	21
3-3	Model architecture for MaskScoringRCNN. The features we use as our embedding are circled in red. . . . .	22
3-4	Segmentations of people and their nearest neighbors. The left most image of each row is the query image. . . . .	24
3-5	Segmentations of cars and their nearest neighbors. The left most image of each row is the query image. . . . .	25
4-1	Main interface of the LabelMeLite annotation tool. . . . .	28
4-2	Main interface of the LabelMeLite annotation tool with the first image of the ADE20K training set loaded. . . . .	29
4-3	AMT Edit annotation interface. . . . .	30
4-4	AMT Yes/No annotation interface. . . . .	31



# List of Tables

3.1	Annotation statistics from the simulated procedure on ADE20K. . . .	23
-----	---	----



# Chapter 1

## Introduction

Segmentation, more specifically instance segmentation, is the problem in computer vision of classifying the pixels of an image into their objects and categories. It is a central problem to many perception tasks in fields like robotics, autonomous driving, and medical imaging.

The current approach involves training neural networks on large annotated datasets. With the rise of deep learning, the last few years have seen a dramatic increase in the size of the best performing models. Increasingly large models require increasingly large training sets to realize their model capacity. Chen et al. [1] finds that performance increases logarithmically based on the size of the training set.

Segmentation data, however, is one of the most expensive types of data to collect. Each annotation typically requires a human annotator to painstakingly trace an object in an image with a polygon. These pixel-level labels quickly become prohibitively expensive to collect. Segmentation datasets today contain on the order of 10-100K images. Classification datasets, on the other hand, regularly contain millions of images. This is because image-level labels are easily scraped off the internet by search engines. Bridging this gap will require heavily automating the segmentation annotation procedure.

In this thesis, we propose using state-of-the-art segmentation networks to generate millions of predictions, clustering and filtering these predictions for segmentations above a certain quality, and then saving those good predictions as our dataset. Using

a segmentation model to generate training data might seem like wishful thinking but our goal is not to generate perfectly clean human-level annotations. Our goal is to maximize the amount of segmentation data we collect per unit of annotation time. The segmentation model here functions as a search engine for objects. By clustering similar predictions together, we ask human annotators a single question and then propagate their answer to nearby instances, amplifying the amount of data we get back per question. Recent work in AI-powered smart annotation tools reduces the effort required to annotate a single instance. In this work, we explore reducing the redundancy in annotating multiple similar instances.

Given an initial segmentation dataset and a large pool of unlabeled images, our goal is to, at very low cost, collect a secondary dataset on the unlabeled images that improves performance on the initial dataset upon re-training. While the segmentation quality on the secondary dataset will be lower but we hope to more than compensate with size.

The rest of thesis are organized as follows. Chapter 2 covers related work. We discuss existing segmentation datasets and other approaches towards scaling them. Chapter 3 discusses our approach, the models and datasets we use, our clustering and filtering methods, and the experiments we run. Chapter 4 details the design and implementation of our annotation tool, LabelMeLite. Chapter 5 concludes with a summary of our contributions and possible future directions.

# Chapter 2

## Related Work

In this chapter, we discuss related work in collecting segmentation datasets. We review existing datasets and we discuss approaches for scaling them with machine learning.

### 2.1 Segmentation Datasets

Datasets define the problems computer vision researchers work on and provide benchmarks with which to measure progress. Much of the progress in computer vision today is thanks to pioneering datasets like Imagenet [2], MNIST [3], and COCO [4]. We review the most relevant segmentation datasets below.

**Pascal VOC [5]** was one of the first segmentation datasets collected. Released in 2009, it contains 10K images with 30K instances over 20 categories and was largely annotated by in-house experts. While this dataset was widely used and oversaw many important developments in segmentation [6] and object detection [7, 8], it has been since been declining in popularity due to its small size.

**COCO [4]** is currently the most popular benchmark for instance segmentation. Released in 2014 by Microsoft Research, it contains 163K images with 1.2M instances over 80 categories. It was collected with a three step annotation process of category labeling, instance spotting, and then segmentation, crowdsourced on Amazon Mechanical Turk (AMT). Due to the crowdsourcing, segmentation quality is not very

high. Nonetheless, this benchmark and its evaluation criteria of average precision (AP) are standard. With a reported annotation time of 80 seconds per instance, we estimate a total annotation cost for the dataset of \$160K.

**ADE20K [9]** represents an extraordinary effort by a single expert annotator to fully annotated a dataset. Released by our lab here at MIT in 2017, this dataset contains 25K images with 450K instances over approximately 3000 categories. When compared to COCO, ADE20K has more complex scenes, higher annotation density, and higher segmentation quality, all resulting in a more challenging benchmark. Most categories however have too few instances for training so the benchmark most commonly associated with this dataset is limited to the 100 most frequent categories. The work in this thesis focuses primarily on scaling this dataset. All images were annotated with the LabelMe interface [10].

**Open Images V5 [11]** is currently the largest segmentation dataset. Released by Google earlier this year, it contains 1M images with 2.8M instances across 350 categories. Annotated with an interactive segmentation model twice as fast as the annotation pipeline for COCO, we estimate they spent a similar amount of money for double the size. See Section 2.2 for more details. While this means segmentation datasets have reached the million image mark, Open Images V5 leaves much to be desired. The dataset contains only 2.8 annotations per image, as opposed to 18 in ADE20K and 7.4 in COCO, and has a strong centering bias. Despite the refined annotation tool, since model performance increases logarithmically with respect to training data [1], this decrease in annotation cost is disappointingly linear.

**Cityscapes [12]**, **Mapillary Vistas [13]**, and **BDD100K [14]** are popular segmentation datasets for self-driving cars. Released in 2016, 2017, and 2018, they contain 5K, 25K, and 100K images respectively. Driven by intense interest in the autonomous vehicle space, this growth was made possible by new annotation tools [15] and data labeling startups. When Cityscapes was first released, they reported an annotation time of 90 mins per image. At \$8/hr, that is \$12 per image. As of 2019, one data labeling startup, Scale, quotes around half that price, \$6.40 per image.



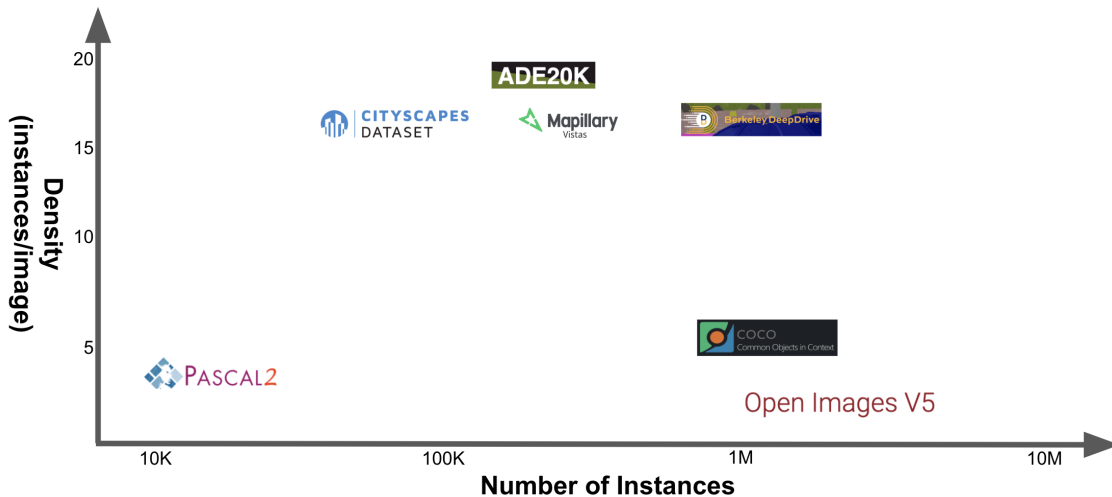


Figure 2-1: Plot of dataset size and annotation density for various segmentation datasets. Both metrics are desirable.

## 2.2 Smart Annotation Tools

While better annotation tools can make a big difference in annotation cost, interactive segmentation models, where humans and machines work together in the annotation process, offer the potential for even more significant time savings. Here we review two such collaborative approaches.

**Polygon RNN [16]** is a semi-automated annotation tool where the human annotator draws a bounding box around an object that the network then uses to predict a segmenting polygon. Where most segmentation models predict pixel labels and output segmentation masks, PolygonRNN predicts polygon vertices that the human annotator can use to grasp and refine the annotation with. While this approach is superior to GrabCut-based approaches [17], this algorithm underperforms state-of-the-art segmentation models and when applied iteratively, often fights the human annotator in the annotation process.

**Open Images V5 [11]** introduces an interactive segmentation model based on corrective clicks. The network produces an initial segmentation mask that the human annotator corrects with inclusive and exclusive clicks. The network then incorporates this information to generate a new mask. This process can be repeated up to four times. They demonstrate that this approach is similar in quality to and twice as fast

as traditional annotation methods and use it to collect a large segmentation dataset.

## 2.3 Semi-supervised Learning

While smart annotation tools are a significant improvement over normal annotation methods, they can only increase annotation speed by a constant factor. This is because they still require human annotators to go over each and every instance. To scale segmentation datasets even further, we take inspiration from semi-supervised learning approaches. Semi-supervised learning is a type of machine learning that combines labeled data with abundant unlabeled data for training deep neural networks.

**Data distillation [18]** is a method for generating training labels on unlabeled data from ensembled network predictions. Radosavovic et al. finds that for human pose detection and object detection, networks trained with these labels in addition to the original labeled data outperform networks trained on labeled data alone. They demonstrate that a model can improve its performance with its own predictions on unlabeled data without any human input. In this work, we explore whether or not we can add a small amount of human input to get an even larger improvement in performance.

**Label propagation [19]** is a technique for propagating labels from labeled examples to their unlabeled neighbors. The manifold assumption claims similar examples with similar embeddings should have the same label. Iscen et al. demonstrates that we can use embeddings from a deep neural network to build a nearest neighbor graph for label propagation and that training with the propagated labels improves performance on several classification tasks, especially in the low-data regime. In this work, we apply a similar method for segmentation and add a human-in-the-loop step for data collection and greater control over the process.

# Chapter 3

## Clustering for Segmentation Data Collection

In this chapter, we discuss our iterative cluster-based approach to segmentation data collection. We begin with an overview of the pipeline. Then, we discuss the segmentation models we use. We test our approach with simulated annotators and show preliminary results on a pool of unlabeled images. Lastly, we discuss next steps towards collecting a real large-scale segmentation dataset.

### 3.1 Overview

The annotation process consists of a segmentation model, an annotator, a segmentation dataset, and a pool of unlabeled images. In this work, our segmentation dataset is the ADE20K [9] dataset and our pool of unlabeled images is the Places [20] dataset. The ADE20K training set contains 20K images while the Places dataset contains 8M images.

First, we train the segmentation model on ADE20K. Then, we run inference on Places and save all the predictions and embeddings. Next, we cluster the predictions by their embeddings with hierarchical clustering and choose a fixed number of representative segmentations from each cluster for annotation. Annotation is a simple yes/no question for whether the segmentation is sufficiently high quality. Specifically,

we ask whether or not the predicted segmentation has an intersection-over-union (IOU) with the ground truth greater than some threshold,  $t_{iou}$ . If the agreement between representative segmentations in a cluster is below another threshold  $t_{agr}$ , we split the cluster until each individual cluster reaches agreement. We use these representative segmentations to accept or reject entire clusters and keep the accepted clusters for our dataset. Lastly, we re-train the segmentation model on our new dataset to measure our progress and repeat the entire process until it stops improving.

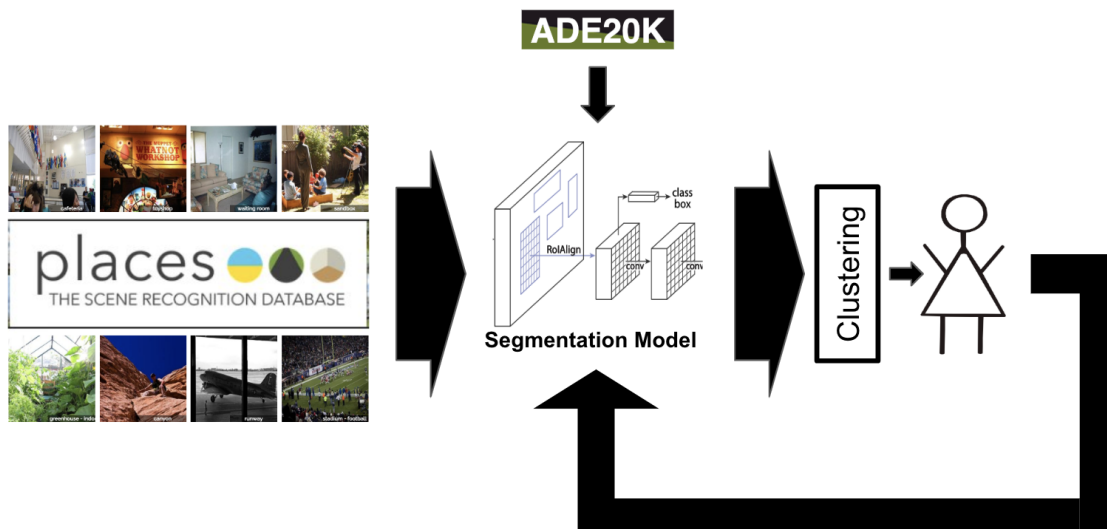


Figure 3-1: Visualization of the annotation pipeline. Clustering prevents the human annotator from being overwhelmed.

## 3.2 Models

This annotation procedure is model agnostic. We can use any model or combination of models to generate segmentation predictions. In this work, we primarily use the two models below.

### 3.2.1 MaskRCNN

MaskRCNN [21] is the current state-of-the-art for instance segmentation. It uses a ResNet [22] backbone for feature extraction and a Region Proposal Network (RPN)

module for object proposals. The object proposals are used in the ROIAlign operation to crop the image features from the backbone into object features. Object features are then fed into two parallel heads. One head predicts object category while the other head predicts segmentation mask.

In our experiments, we use MaskRCNN with a ResNet101-FPN backbone as our segmentation model and save any prediction with a classification score greater than 0.5. We take the 1028D vector from the last fully connected layer of the classification head as our embedding. The hope is that with this embedding, objects with good segmentations will be clustered with other objects with good segmentations. Objects with bad segmentations will go somewhere random in embedding space or form bad clusters. Classification score is somewhat correlated with segmentation quality so this should be reflected in the embedding.

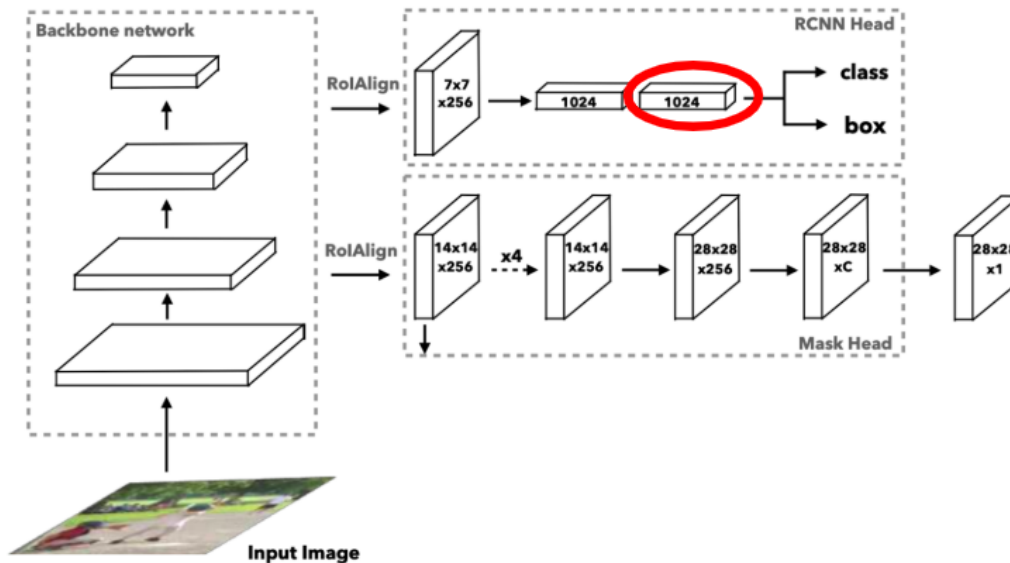


Figure 3-2: Model architecture for MaskRCNN. The features we use as our embedding are circled in red.

### 3.2.2 MaskScoringRCNN

The classification head in MaskRCNN however, does not ever see the predicted segmentation. The two model heads are parallel. This means the embedding from the classification head does not actually reflect segmentation mask and that the corre-

lation between classification score and segmentation quality is coincidental. Author et al. [23] finds that classification score in MaskRCNN is not well correlated with segmentation quality.

MaskScoringRCNN [23] resolves this by adding an additional head to MaskRCNN explicitly for predicting segmentation quality by regressing the IOU of the predicted segmentation. This head concatenates the segmentation mask with the object features before doing the regression. We take the last layer of the classification head and the last layer of the MaskIOU head as our 2048D embedding. This should be a better embedding and drop in replacement for MaskRCNN. However, since we have not fully implemented it in our annotation pipeline yet, the rest of this work will use MaskRCNN.

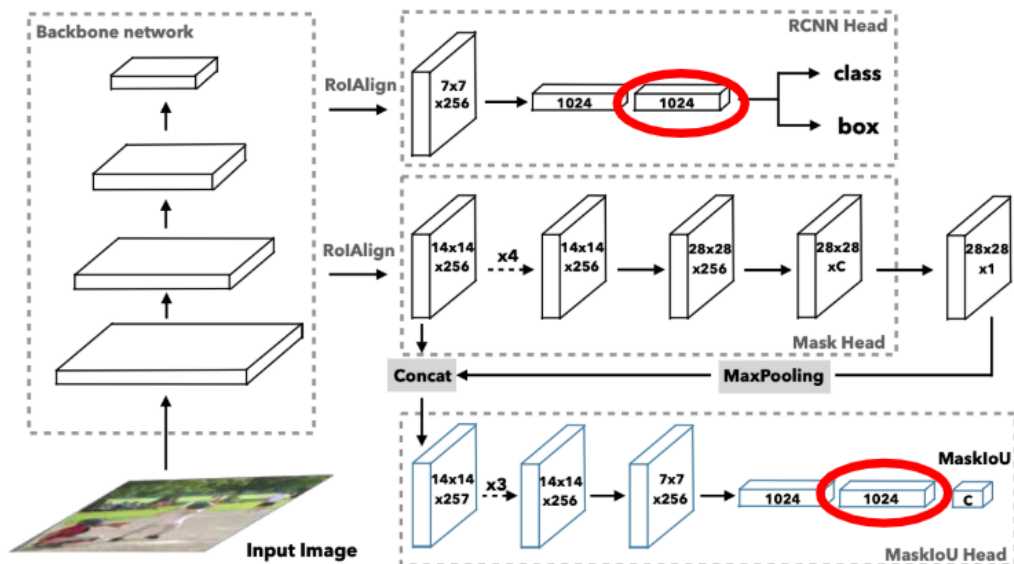


Figure 3-3: Model architecture for MaskScoringRCNN. The features we use as our embedding are circled in red.

### 3.3 Simulation on ADE20K

Before deploying the pipeline on Places, we first tested our annotation procedure with a toy experiment on ADE20K. We split ADE20K into two halves, SplitA and SplitB, and attempted to use SplitA to annotate SplitB. SplitA was our initial segmentation

dataset and SplitB was our pool of unlabeled images. We used the ground truth for SplitB to answer the yes/no questions that we would have otherwise asked human annotators. We accepted any segmentation with IOU greater than 0.8.

First, we trained a segmentation model on SplitA. This got an average precision (AP) of 17.02 on the ADE20K validation set. Then, we ran inference on SplitB. This produced 82K predictions. The 82K predictions however was not enough to form for useful clusters. Thus, we decided to omit the clustering and simulated yes/no questions for each predicted segmentation independently. While this is probably made the annotation procedure not cost-effective, we wanted to see whether the pipeline, in this best case scenario, could collect a segmentation dataset and improve network performance.

In the first iteration, we asked 82K yes/no questions and accepted 21K segmentations. Finetuning with the new segmentations increased the AP on the validation set to 17.79. Then, we ran inference on split B again and produced another 82K segmentations. Removing duplicates with the previous iteration brought this down to 33K segmentations. In the second iteration, we asked 33K yes/no questions and accepted 2.5K segmentations. We repeated this until we no longer discovered any more new objects.

Iteration	Questions Asked	Segmentations Collected	Model Performance (AP)
0	-	-	17.08
1	82306	21034	17.79
2	32928	2503	17.93
3	32213	1414	17.79
4	30677	956	17.85
5	29727	646	17.69
6	29368	645	17.85

Table 3.1: Annotation statistics from the simulated procedure on ADE20K.

After six iterations, we collected 27K near-perfect segmentations or 28% of all available annotations in SplitB. We successfully used a network’s own predicted segmentations in training and increased the AP of the network from 17.02 to 17.85. Training with the full ADE20K dataset gives a AP of 19.90 but that is the upper

bound on our procedure. When we move to Places, that will be our lower bound.

### 3.4 Clustering in Places

When annotating Places, the clustering step we skipped in simulation becomes essential. The 8M images in Places generate 46M segmentations. It is not feasible to annotate even 0.1% of them. With this many predictions, the clusters we create contain hundreds to thousands of segmentations. We need each cluster to be as pure and consistent with itself as possible.

To gain an intuition for the quality of our clusters, we visualized some of the segmentations and their nearest neighbors below. All clustering and visualization was done on a random 100K image subset of Places. Since increasing the number of images increases our sampling frequency of the embedding space, any patterns we notice here should be more pronounced on the full set of predictions.



Figure 3-4: Segmentations of people and their nearest neighbors. The left most image of each row is the query image.

Figure 3-4 visualizes segmentations of people and their nearest neighbors. Notice that not only are all nearest neighbors also people, they are people in similar poses



or situations. The first row consists of only people occupying a large central portion of the frame with their upper body. The second row consists of only people seated at a dining table. The third row consists mostly of people holding a guitar. The last row consists of people with only a tiny portion of their head visible. Segmentations split up into recognizable subcategories beyond their predicted category.

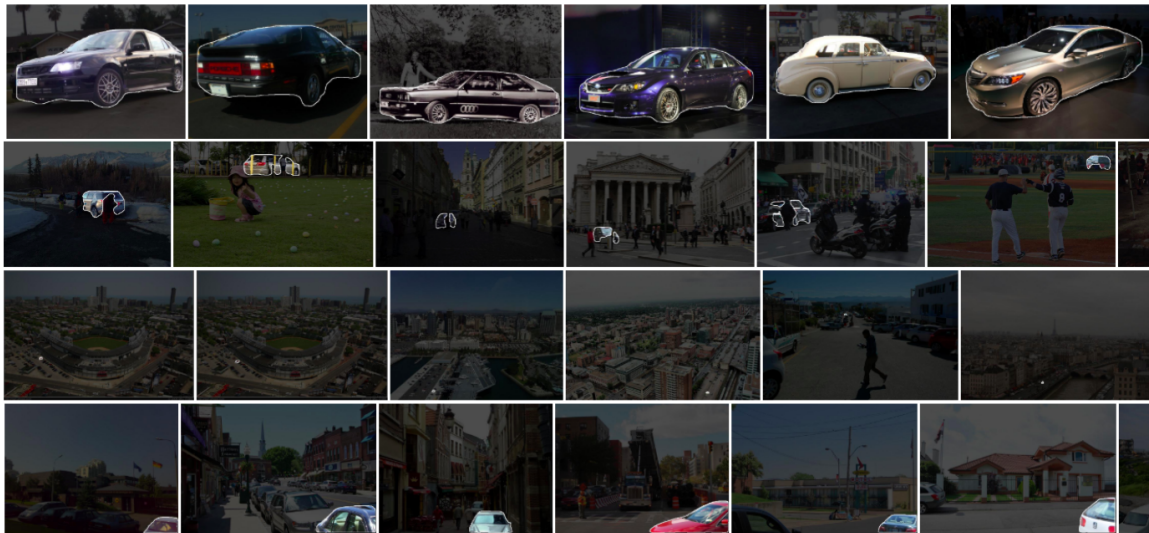


Figure 3-5: Segmentations of cars and their nearest neighbors. The left most image of each row is the query image.

Figure 3-5 visualizes segmentations of cars and their nearest neighbors. The first row consists of large centralized cars. The second row consists of small cars occluded in the middle. The third row consists of tiny cars in the distance. The last row consists of cars in the bottom right corner of an image.

In general, clusters look like they have very strong consistency but without ground truth, this consistency is hard to measure.

### 3.5 Next Steps

With all the predictions and embeddings already generated, the problem of collecting a segmentation dataset on Places essentially boils down to selecting a subset of those predictions that improves network performance. The goal is to extract this subset as cheaply as possible. The simulation in ADE20k showed us that most of the perfor-

mance increase from the annotation procedure comes from segmentations collected in iteration one. This means we should be able to collect a good dataset on Places without iterating.

The next step, before resorting the human annotators, is to experiment with using another segmentation dataset to query for this subset. If we run inference on a segmentation dataset like COCO, we can collect predictions and embeddings for segmentations with IOU greater than 0.8. If we use these predictions to query into the embedding space and retrieve nearest neighbors to those good predictions from Places, we might be able to build a small segmentation dataset that already improves network performance at no cost. The performance change in the segmentation network after retraining could also help us measure the quality of the embedding we used to collect those predictions.

## 3.6 Future Work

After collecting a small segmentation dataset on Places that improves performance on ADE20K, the major challenge will be boosting our coverage of the Places dataset. Executing the full annotation pipeline with hierarchical clustering, iteration, and human annotators will require much more work and will likely have many unforeseen problems. Nonetheless, we believe this approach to segmentation data collection is one that can scale with the enormous appetite of the deep learning algorithms. We will continue working in this direction.

# Chapter 4

## LabelMeLite

In this chapter, we discuss our new annotation tool, LabelMeLite. This tool is designed to be a lightweight modern version of the popular LabelMe tool [10] released in 2008. Our work on this tool predates our work on clustering. Thus, in many ways, this tool is unrelated to the clustering work discussed in previous chapters. Nonetheless, there are many features this tool has that the clustering project might find useful when it reaches the data collection phase. LabelMeLite is now live and the code will be made available shortly.

### 4.1 Motivation

The original goal of this project was to scale segmentation datasets by any means necessary. We decided to create an annotation tool that could edit an segmentation mask from a model as quickly as possible. Existing annotation tools could only support polygons and were designed for annotating images from scratch. We wanted a tool that was Paint-like and supported masks, free-form editing, and intuitive image navigation.

Implementing these features into LabelMe’s aging stack would have been unwieldy. Instead, we saw this as an opportunity to take advantage of new web technologies like HTML5, NodeJS, and Javascript graphics libraries to release a lightweight standalone in-browser image annotation tool.

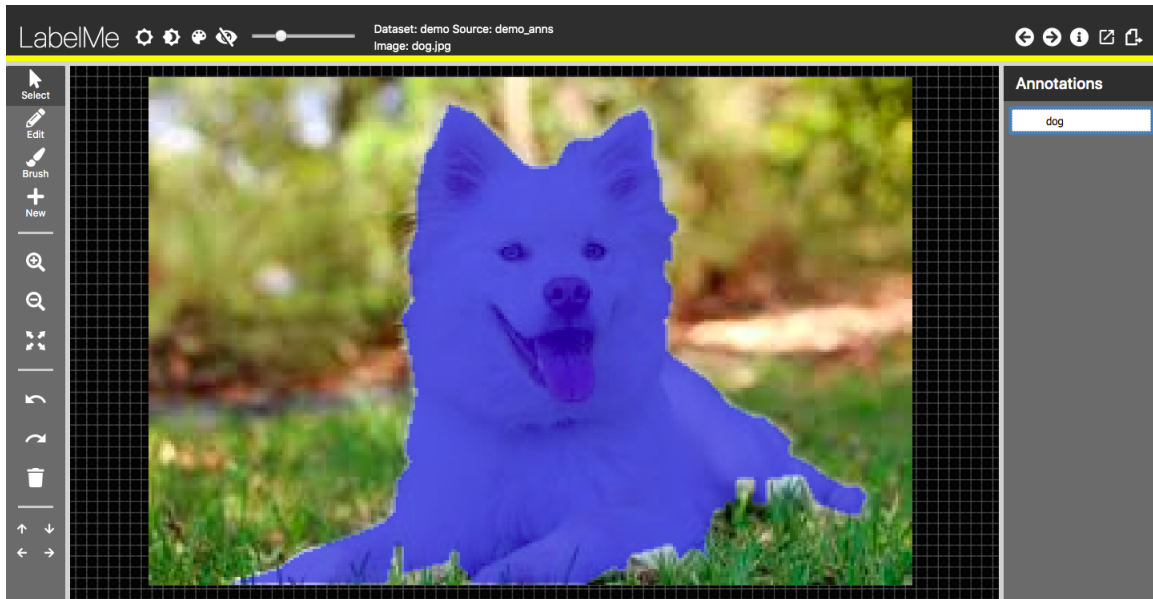


Figure 4-1: Main interface of the LabelMeLite annotation tool.

## 4.2 Main Interface

LabelMeLite loads the image onto a clean black background for annotation. The canvas supports Google Maps-like controls for zooming and navigation. The left and top bars display available tools and options. The right bar displays current annotations.

The default tool is the Select tool. From here, you can navigate the image or double click on an annotation. If there are no annotations to click on, you can select the New tool from the toolbar.

The New tool allows you to draw a new annotation with a polygon. When you close the annotation, the tool prompts you for a category name. When you finish, the annotation is instantly rasterizes to the resolution of the image and creates the corresponding pixel-accurate boundary. The entire tool is mask-based. This allows us to support annotations with holes and sidestep problems with complex self-intersecting polygons.

If you want to edit the annotation, select the Edit tool or double click on the annotation. If you double clicked, the tool will automatically center the annotation on screen for you. The Edit tool is a boundary reroute tool. Edit the annotation by

tracing out a new path. The tool will automatically route the boundary through the new path for you. More instructions are on the website. LabelMeLite also supports a Brush tool and features like undo, redo, delete, and hide.

Existing datasets are easy ported in for visualization and annotation. To load an dataset or your own images, follow the instructions posted on Github. Images and annotations can be served from your own local folder or online.

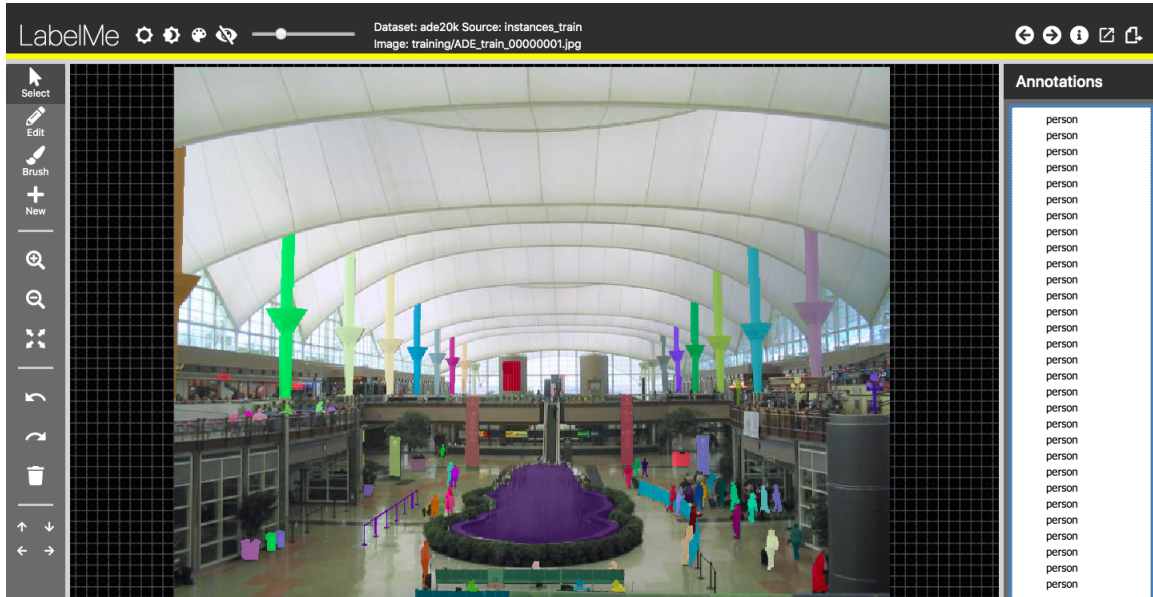


Figure 4-2: Main interface of the LabelMeLite annotation tool with the first image of the ADE20K training set loaded.

## 4.3 Implementation

LabelMeLite is a standalone NodeJS application that serves the annotation tool as a single webpage. Most of the functionality is made possible by the Paper.js vector graphics scripting library and OpenCV.js. Implementing the annotation tool was an exercise in patience, iteration, UX, and web design.

### 4.3.1 COCO API

The most important feature of the annotation tool is our Javascript implementation of the COCO API. The COCO format is a compact convenient way of storing seg-

mentation masks. It uses an encoding scheme similar to LEB128 but with 6 bits and ASCII characters on the run-length encoding of the segmentation mask. With our implementation, we can deserialize and visualize the masks in-browser. This is critical for any large-scale segmentation annotation project. The Places dataset is already a massive dataset. Visualizing each segmentation before sending it for annotation would be incredibly storage and bandwidth intensive.

## 4.4 Amazon Mechanical Turk

Aside from the full interface, we also designed two interfaces specifically crowdsourcing annotations on Amazon Mechanical Turk (AMT).

### 4.4.1 Edit Interface

For editing a single segmentation on AMT, the full annotation tool has too many tools and options. To create a more intuitive interface, we disabled most of the features and turned on the Edit tool and auto-zoom by default. The result was an image with an annotation that looked static yet was editable. Although we have only tested the interface in-house, we are reasonably confident the tool is easy to use.

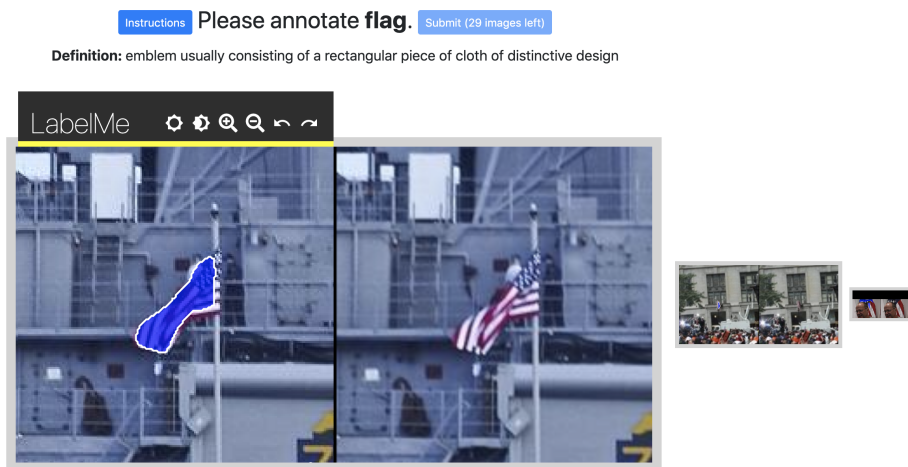


Figure 4-3: AMT Edit annotation interface.

## 4.4.2 Yes/No Interface

For the yes/no annotations described in the previous chapter, we designed an interface very similar to the interfaces used to build large image classification datasets. However, instead of loading static images, we used our Javascript implementation of the COCO API to visualize segmentations in browser.

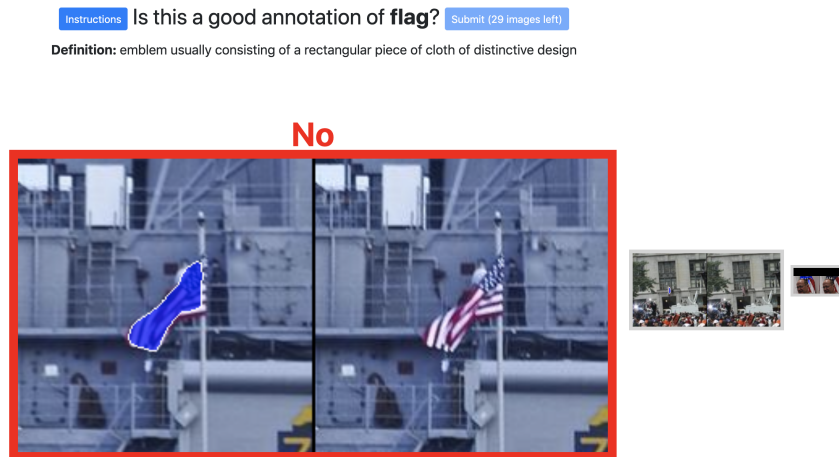


Figure 4-4: AMT Yes/No annotation interface.

## 4.5 Future Work

While majority of the annotation functionality has been implemented, this tool could benefit greatly from navigation and dataset management pages to improve user experience. However, due to the release of a number of other annotation tools [11, 15] this last year, this is not a high priority. We will release LabelMeLite as is and maintain it with respect to the annotation needs of our other projects.





# Chapter 5

## Conclusion

In this thesis, we take on the decades-old challenge of building a large dataset. In doing so, we make the following two main contributions:

1. We present a novel, scalable, cluster-based approach to segmentation data collection. We show promising but preliminary results and lay out next steps.
2. We release a new light-weight annotation tool for editing existing annotations rather than drawing new ones from scratch.

Dataset collection in the future will involve fundamental cooperation between humans and machine learning algorithms. This cooperation will go beyond simply better and faster annotation tools. There is a need for new methods that can direct human annotation effort towards the weaknesses in state-of-the-art models and away from situations they already do well in. Our work is a step in that direction.



# Bibliography

- [1] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. *CoRR*, abs/1707.02968, 2017.
- [2] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein, Alexander C. Berg, and Fei-Fei Li. Imagenet large scale visual recognition challenge. *CoRR*, abs/1409.0575, 2014.
- [3] Corinna Cortes Yann LeCun and Christopher J.C. Burges. The mnist database of handwritten digits. <http://yann.lecun.com/exdb/mnist/>, 1998.
- [4] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.
- [5] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, January 2015.
- [6] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *CoRR*, abs/1411.4038, 2014.
- [7] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
- [8] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
- [9] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *CoRR*, abs/1608.05442, 2016.
- [10] Murphy KP Freeman WT Russell BC, Torralba A. Labelme: a database and web-based tool for image annotation. *International Journal of Computer Vision*, 2008.

- [11] Rodrigo Benenson, Stefan Popov, and Vittorio Ferrari. Large-scale interactive object segmentation with human annotators. *CoRR*, abs/1903.10830, 2019.
- [12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. *CoRR*, abs/1604.01685, 2016.
- [13] Gerhard Neuhold, Tobias Ollmann, Samuel Rota Bulò, and Peter Kotschieder. The mapillary vistas dataset for semantic understanding of street scenes. 2017.
- [14] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. BDD100K: A diverse driving video database with scalable annotation tooling. *CoRR*, abs/1805.04687, 2018.
- [15] Berkeley DeepDrive. Scalabel: A scalable open-sourced annotation web tool. <https://www.scalabel.ai/>, 2017.
- [16] Lluís Castrejón, Kaustav Kundu, Raquel Urtasun, and Sanja Fidler. Annotating object instances with a polygon-rnn. *CoRR*, abs/1704.05548, 2017.
- [17] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. "grabcut": Interactive foreground extraction using iterated graph cuts. *ACM Trans. Graph.*, 23(3):309–314, August 2004.
- [18] Ilija Radosavovic, Piotr Dollár, Ross B. Girshick, Georgia Gkioxari, and Kaiming He. Data distillation: Towards omni-supervised learning. *CoRR*, abs/1712.04440, 2017.
- [19] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. Label propagation for deep semi-supervised learning. *CoRR*, abs/1904.04717, 2019.
- [20] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Antonio Torralba, and Aude Oliva. Places: An image database for deep scene understanding. *CoRR*, abs/1610.02055, 2016.
- [21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.
- [23] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring R-CNN. *CoRR*, abs/1903.00241, 2019.