Where are the objects? Weakly Supervised Methods for Counting, Localization and Segmentation

by

Issam Laradji

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

DOCTOR OF PHILOSOPHY

in

The Faculty of Graduate and Postdoctoral Studies

(Computer Science)

THE UNIVERSITY OF BRITISH COLUMBIA

(Vancouver)

April 2020

The following individuals certify that they have read, and recommend to the Faculty of Graduate and Postdoctoral Studies for acceptance, the dissertation entitled:

Where are the objects? weakly supervised methods for counting, localization and segmentation

submitted by Issam Hadj Laradji in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Computer Science.

Examining Committee:

Mark Schmidt, Associate Professor, Computer Science, UBC

Supervisor

David Poole, Professor, Computer Science, UBC

University Examiner

Zhen Jane Wang, Professor, Electrical and Computer Engineering, UBC

University Examiner

Leonid Sigal, Associate Professor , Computer Science, UBC

Supervisory Committee Member

Jim Little, Professor, Computer Science, UBC

Supervisory Committee Member

Yuri Boykov, Professor, Computer Science, The University of Waterloo

External Examiner

Abstract

In 2012, deep learning made a major comeback. Deep learning started breaking records in many machine learning benchmarks, especially those in the field of computer vision. By integrating deep learning, machine learning methods have became more practical for many applications like object counting, detection, or segmentation. Unfortunately, in the typical supervised learning setting, deep learning methods might require a lot of labeled data that are costly to acquire. For instance, in the case of acquiring segmentation labels, the annotator has to label each pixel in order to draw a mask over each object and get the background regions. In fact, each image in the CityScapes dataset took around 1.5 hours to label. Further, to achieve high accuracy, we might need millions of such images.

In this work, we propose four weakly supervised methods. They only require labels that are cheap to collect, yet they perform well in practice. We devised an experimental setup for each proposed method. In the first setup, the model needs to learn to count objects from point annotations. In the second setup, the model needs to learn to segment objects from point annotations. In the third setup, the model needs to segment objects from image level annotations. In the final setup, the model needs to learn to detect objects using counts only. For each of these setups the proposed method achieves state-of-the-art results in its respective benchmark. Interestingly, our methods are not much worse than fully supervised methods. This is despite their training labels being significantly cheaper to acquire than for the fully supervised case. In fact, in fixing the time budget for collecting annotations, our models performed much better than fully supervised methods. This suggests that carefully designed models can effectively learn from data labeled with low human effort.

Lay Summary

Deep learning has been shown to be useful in many everyday systems. Unfortunately, many systems based on deep learning need a lot of labels to work well. Acquiring these labels requires a lot of human effort which can be an impractical endeavor. As a result, new deep learning systems came out that only need few labels to work fairly well, but not good enough for real-life applications. In this work, we develop new state-of-the-art systems to handle three types of applications in image processing. These applications are counting, finding objects in images and drawing their boundaries. Our systems only need labels that are very cheap to acquire and were shown to achieve good results in many standard benchmarks.

Preface

I am the primary contributor for each of the following 4 papers which I included as Chapter 2-5 in my thesis. I have came up with the idea, conducted the experiments and wrote the papers. The co-authors guided me in terms of what experiments are required and how I should write the papers.

- Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vazquez, Mark Schmidt. "Where are the blobs: Counting by localization with point supervision." ECCV 2018.
- Issam H. Laradji, Negar Rostamzadeh, Pedro O. Pinheiro, David Vazquez, Mark Schmidt. "Instance Segmentation with Point Supervision.".
- Issam H. Laradji, David Vazquez, Mark Schmidt. "Where are the Masks: Instance Segmentation with Image-level Supervision." BMVC2019.
- Issam H. Laradji, David Vazquez, Mark Schmidt. "Object Localization for Dense Scenes with Count Supervision".

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my family, friends, and supervisors who have given me the most help during the course of my graduate work.

Chapter 1

Introduction

1.1 The Rise of Deep Learning: Recent Advances

Deep learning began to receive a renewed interest when it achieved a record-high classification score on the ImageNet challenge [31]. This result showed that deep learning could effectively learn from a huge amount of images that represent diverse, real-life, everyday scenes. It also showed that deep learning can extract strong features from images. As a result, deep learning has become successful for a wide range of large-scale computer vision applications. Examples include keeping track of crowd counts in surveillance cameras [89, 29], monitoring species counts [8], and helping autonomous cars navigate urban roads [26].

As it became clearer that deep learning was an effective tool, the research community shifted its focus towards using deep learning for many classic machine learning tasks. These tasks include data analytics [85], image segmentation [49], text summarization [22], and video understanding [133]. Communities outside machine learning also became more interested in deep learning. More effort was put into investigating the efficacy of deep learning in areas like medicine [23] and algorithmic trading [32].

Deep learning has made several research areas more prominent, including, style transfer [54] and adversarial learning [43]. This is because deep networks seem to capture image semantics at different levels of abstraction [50]. Compared to hand-engineered features [28], the deep network features make a much more powerful representation of the data. In fact, features extracted from a trained network can be transferred from one domain to another [80]. In contrast, hand-engineered feature extraction methods [56] need to be tuned towards one dataset of limited size.

Such powerful data representations are useful for tasks like style transfer. The idea behind style transfer is to change the style of a target image using a source image. For example, prior to deep learning, style transfer methods relied on Gaussian-based algorithms [3, 46]. However, this led to limited success due to the restricted capacity of these methods. With deep methods pretrained on ImageNet [118], styles between images can be efficiently transferred across regions [55]. This made styling images with sophisticated art require much less human effort. For adversarial learning, the task is to train a model using malicious inputs designed to fool the

model. The implication is that the trained model can be used in image-to-image translation [52], image generations [44], and adversarial defense [108]. Similar to style transfer, models in this framework require the capacity to learn complex representations, which are given by deep learning networks.

Different industries [37, 98, 107] have become more interested in adopting deep learning in their applications. As a result, many startups in this field have acquired large amounts of funding.[1] This led to a huge number of deep learning based work being submitted to machine learning conferences.[2] As a result, many new real-life applications emerged.[3] Fortunately, a deep learning method trained on one dataset can be effectively used for other datasets where the objects of interest are the same. For instance, a person detector trained on COCO [75] can be deployed to detect persons in new scenes.

Unfortunately, many of these applications include classes that are different from standard ones. Such classes include penguins, cells, and plants. Thus, new datasets need to be collected and annotated, in order to fine tune the model for the new classes. The main challenge, however, is the annotation cost. Deep learning methods require huge amounts of data to train properly. This annotation requires costly human effort. For tasks such as semantic segmentation, it can take 1.5 hours to annotate a single image [26].

In most computer vision problems, we need many annotated data because models need to learn the variability of the objects in terms of shape, size, pose, and appearance. Objects may appear at different angles and resolutions, and may be partially occluded. Further, the background, weather conditions, and illuminations can vary widely across scenes. Therefore, the model needs to be robust enough to recognize objects in the presence of these variations.

Thus, enormous interest has been put forth towards creating deep learning methods that can learn from labels that are cheap to collect [120, 12, 90, 60, 1, 109, 57, 141, 24]. These are known as weakly supervised methods. In most cases, weakly supervised methods perform much worse than fully-supervised methods. This presents a dilemma. However, the cost of collecting a fully supervised training set might be higher than simply manually labeling the test set. Thus, collecting such training set is not a wise decision. On the other hand, weakly supervised methods might require much less human effort to achieve reasonable performance.

In this work, we propose weakly supervised methods in order to close the performance gap between weakly and fully-supervised methods. In the next sections,

[1] https://www.forbes.com/sites/parmyolson/2019/03/04/nearly-half-of-all-ai-startups-are-cashing-in-on-hype/#7ca745afd022

[2] https://medium.com/syncedreview/paper-submissions-break-neurips-2019-paper-submission-system-884a60e32a82

[3] https://www.kaggle.com/

we describe three fundamental tasks in computer vision. They are object counting, instance segmentation and crowd localization. We will illustrate their importance in real-life applications, explain relevant existing methods, and describe their relationship to my PhD work. We also discuss the limitations of current methods, and propose novel methods that address their limitations.

1.2 Background

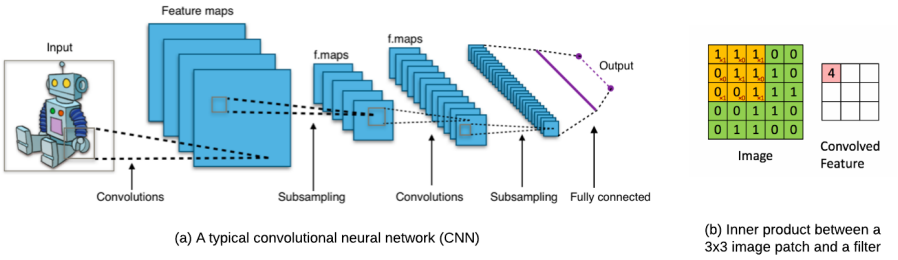


Figure 1.1: **Convolutional neural networks.** An architecture consisting of a set of layers that extract features at different levels of abstraction for an image (*Link: https://i.stack.imgur.com/3tUPW.png.*).

Convolutional Neural Networks. Convolutional neural networks (CNNs) are a type of a neural network architecture that has become standard for image classification. They are inspired by Fukushima [39], and were successfully applied to optical character recognition in 1998 [67]. The main component of a CNN is the convolutional layer. This layer consists of a set of rectangular matrices, known as filters, that are used to extract features from an image. Such filters are often small (usually 3x3) which is used to extract features at every (3x3) patch of an image. This process is known as convolution. To extract these features for a single patch, the inner product is computed between the filter and the patch (Figure 1.1). The result is placed as a single entry in a new feature map. Thus the feature map represents features extracted at different locations of the image. One property of the convolution operation is that it is translation equivariant. If the same object appears at different positions, the output of the filter used in the convolution is the same at each of those positions.

CNNs can be made deeper by adding more layers. For instance, another set of filters can be applied to each patch of the feature map. In turn, a new set of features get extracted for each spatial location of that feature map. The advantage

of having these multiple layers of features is that they might represent higher level of semantics. For example, if the first layer features represent low-level features like edges, then the second layer features can extract more complex details like contours that combine those edges. Since the filters are typically of size 3x3, they can only capture smaller objects in the image. In order to capture larger objects, the feature maps are downsampled before the next convolutional layer. This makes larger objects look smaller so that their features can be extracted by the filters. Another advantage of downsampling feature maps is translation invariance. An important feature can be captured even if it is at a different spatial position. This downsampling process is called max-pooling.

This framework is important for image classification. Higher level features are often necessary for classifying images correctly. Successful CNNs have many layers (up to 152 [48]) and have consistently been considered as state-of-the-art for image-classification.

In image classification, the last feature map of the CNN is flattened into a single long vector. The weighted sum of this vector is used to obtain a probability that the image belongs to object classes.

CNNs are trained by adjusting the values of the filters and the parameters used to compute the last feature maps. The goal is to maximize the image classification accuracy on the training set. Since CNNs achieved a lot of success for image classification with large-scale datasets, they form the foundation for many computer vision solutions.

Fully Convolutional Neural Networks. CNNs can be used for tasks other than image classification as well. These tasks include image segmentation and object detection. For segmentation, a CNN model can classify each pixel in an image in order to obtain the mask for each object and the mask for the background region.

This type of CNN is known as Fully convolutional neural networks [79] (FCNs). FCNs do not have a fully connected layer as the last layer like with CNNs that are used for classification. FCN layers are mostly convolutional layers that output a series of feature maps. The last feature map represents a classification score at every spatial location of the image. The depth of that feature map is the number of classes of the dataset. In order to obtain the label for a single pixel, the entry with the largest value across the class dimension is set as the label. The first FCN based method was proposed by Long et al. [79], which was successfully applied for large scale semantic segmentation (see Figure 1.2).

Instance Segmentation. The task of instance segmentation is to get the mask for each object in the image and classify them. A gold standard method for this task is mask region-based convolutional neural networks (Mask R-CNN) [49]. Mask R-CNN performs instance segmentation in two stages. In the first stage, a region

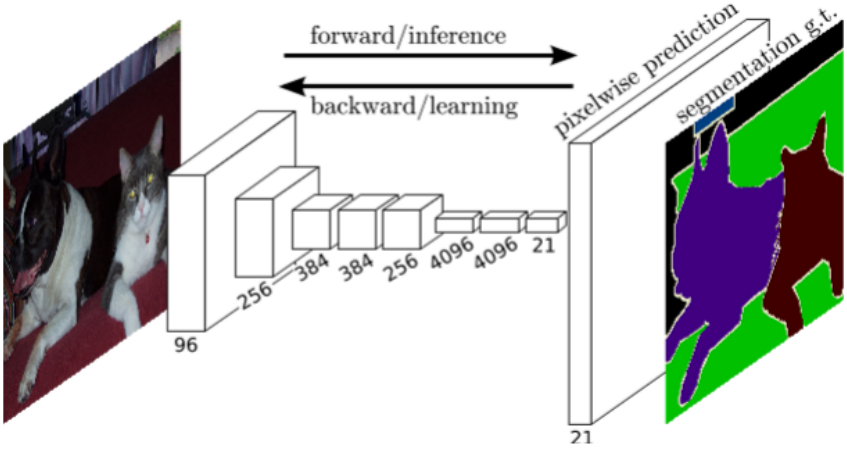


Figure 1.2: A fully convolutional neural networks classifying each pixel with the object category. Image obtained from Long et al. [79].

proposal network outputs a set of bounding boxes (usually a thousand) indicating possible locations for the objects of interest (Figure 1.3). The object in a bounding box can be one of many categories. It can be a rock, dog, horse, pedestrian and so on. In the second stage, a feature vector and a feature map are extracted for each proposed bounding box. The feature vector is used to classify the bounding box, while the feature map is used to segment the bounding box. For classification, a set of fully connected layers is applied to the feature vector to obtain the classification score. The size of the feature vector for each bounding box is the same, and they are usually 1024. For the segmentation, the bounding box feature map is passed through a fully convolutional neural network to classify each pixel in that bounding box. The pixels classified as foreground define the mask for the object within the bounding box.

Many instance segmentation methods build upon mask R-CNN such as MaskLab [18]. Also, faster methods than mask R-CNN emerged that can perform instance segmentation in real-time [38]. These methods are known as single-stage detectors which skip the first stage of mask R-CNN by not requiring the extraction of candidate bounding boxes.

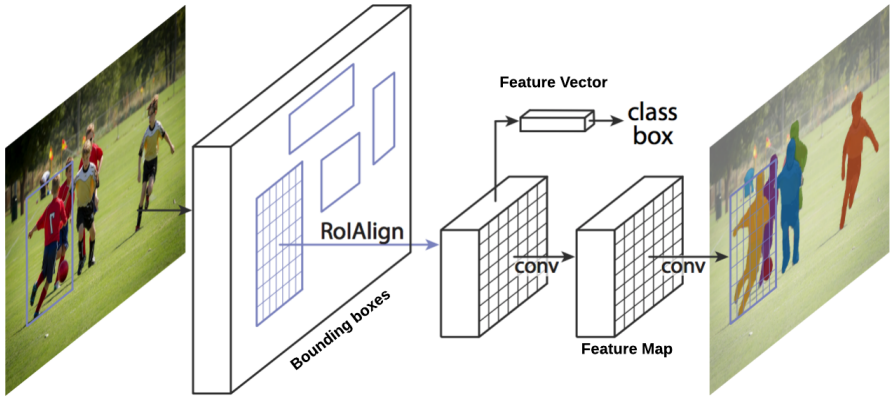


Figure 1.3: **Instance Segmentation** Mask R-CNN is a popular framework for instance segmentation. It is a two-stage detector that generates bounding boxes, classifies them, and segments them. Image obtained from He et al. [49].

1.3 Object Counting

Object counting, as its name would suggest, is the task of counting the number of objects in an image. It's widely used for counting crowds of objects from a fixed camera, which plays an important role for surveillance systems [128, 132], traffic monitoring [29, 45], ecological surveys [8], and cell counting [25, 68]. In traffic monitoring, counting methods can be used to track the number of moving cars, pedestrians, and parked cars. They can also be used to monitor the count of different species such as penguins, which is important for animal conservation. Furthermore, object counting methods have been used for everyday scenes to count objects coming from a large number of classes [35].

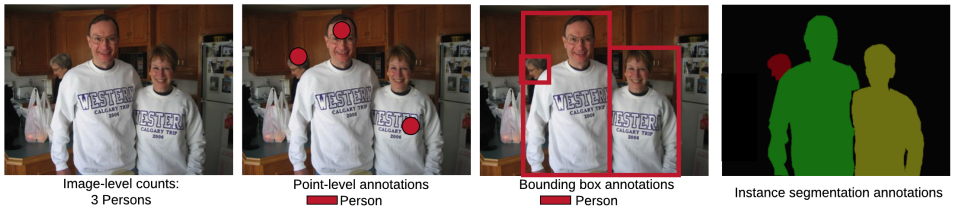


Figure 1.4: Various types of annotations. From left to right, the annotations are ordered in increasing level of difficulty. Image obtained from the PASCAL PART dataset [20]

A straightforward way to teach a model to count is to train a model to detect bounding boxes [103]. However, labeling images with bounding boxes does not come cheap. In fact, it takes time and mental focus to obtain the right width and height of the objects, especially when they overlap [57]. This challenge applies to training models to perform the task of object detection as well. Thus, it is important to consider weaker supervision [16]. In Figure 1.4, we show different annotations starting from the weakest labels from the left to the strongest labels on the right. Image-level labels only indicate whether certain categories of objects exist in the image. Image-level counts requires the annotator to count the number of objects existing in the image for each category. Point-level annotations requires a single pixel being annotated for each object. Bounding box annotations requires carefully placed boxes that has the exact same height, and width as the object within it. Finally, instance segmentation labels require per-pixel labels that distinguish between the object classes and instances.

Early methods relied on bounding box annotations to count the number of objects in the image [70]. Their frameworks combine support vector machines and histogram of gradient features to detect the objects and then count them [28, 127, 122]. Such methods have fallen out of fashion in favor of methods that learn from point-level annotations, labels that are less costly to acquire than bounding boxes [10].

For counting with point-level supervision, perhaps the first method was proposed by Lempitsky and Zisserman [68]. It is a density-based method that explicitly learns how to count rather than to detect objects. They transform the point-level annotations into a density map using a Gaussian kernel. Then, they train their model using a least-squares objective to predict the density map. Many methods build on top of this framework by incorporating different deep learning architectures [89, 5, 6, 7, 45, 135, 71]. Nonetheless, the training procedures are similar between them.

Interestingly, density-based methods outperform detection-based methods for counting [71], despite needing less supervision. The intuition is that detection-based methods try to learn a more difficult task, which includes predicting the location, size, and shape of each object. As a result, this can make the model worse at counting.

On the other hand, density-based methods face a common issue: they assume a fixed object size and shape. The point annotations are converted to a density map by applying a Gaussian kernel with a fixed size [45]. As a consequence, it becomes more difficult for the model to distinguish between objects of different sizes and shapes.

We address this limitation by proposing a localization-based fully convolutional neural network (LCFCN) [65]. Our counting method uses the provided point

annotations to guide its attention to the object instances in the scenes in order to learn to localize them. It has a novel loss function that encourages the model to output instance regions such that each region contains a single object instance (in other words, a single point-level annotation). Similar to detection, the predicted count is the number of predicted instance regions. As a result, our model has the flexibility to predict different sized regions for different object instances, which makes it suitable for counting objects that vary in size and shape. Another advantage of using LCFCN is obtaining object locations. These locations are around the center of each object which are better than a density heatmap which does not discriminate between overlapping objects.

At the time of submission to ECCV2018, LCFCN achieved new state-of-the-art results. This phenomenon seems to contradict with the statement made earlier that solving a harder task, such as detection, results in poorer performance for easier tasks like counting. After all, LCFCN is a localization method which is not as difficult as the detection task, but not as easy as the counting task. Our intuition, however, is that having the model explicitly learn to locate the objects is necessary for counting objects, much like what humans do when they count.

The contributions of LCFCN [65] are as follows: (1) we propose a novel loss function that encourages the network to output a single blob per object instance using point-level annotations only; (2) we design two methods for splitting large predicted blobs between object instances; and (3) we show that our method achieves new state-of-the-art results on several challenging datasets including the Pascal VOC and the Penguins dataset.

1.4 Instance Segmentation

Instance segmentation is the task of classifying every object pixel into a category and discriminating between individual object instances. It has a wide variety of applications such as autonomous driving [26], scene understanding [75, 35], and medical imaging [94].

Most instance segmentation methods follow a two step procedure [49, 18, 38], where they first detect objects and then segment them. For instance, Mask R-CNN [49] uses Faster R-CNN [103] for detection and an FCN network [79] for segmentation. However, these methods require dense labels which leads to a high annotation time for new applications.

Another class of instance segmentation methods obtain the object masks by grouping pixels based on a similarity measure. Notable works in this category include methods based on watershed [9], template matching [123] and associative embedding [86]. Fathi et al. [36] proposed a grouping-based method that first

learns the object locations and then learns the pixel embeddings in order to distinguish between object instances. These methods also require per-pixel labels which are costly to acquire for new applications.

Many weakly supervised methods emerged to overcome the need for per-pixel labels. Instead, they only require weaker labels including bounding boxes [57], scribbles [74], and image-level [141, 24] annotations. This makes acquiring datasets a significantly more scalable endeavour. According to Bearman et al. [10], it requires 20 sec/img to acquire image-level labels (which are labels that only indicate whether an object class appears in an image) for PASCAL VOC [35], compared to 239.7 sec/img for acquiring per-pixel labels. To date, only two weakly supervised methods address instance segmentation with image-level labels [141, 24]. Unfortunately, these two methods have been only shown to work on the most basic instance segmentation dataset, PASCAL VOC 2012, and their results are much worse than fully supervised methods. To address these limitations, we propose two methods, one that learns from point-level annotations and the other from image-level annotations.

Our method, WISE-Net, achieved new state-of-the-art results for instance segmentation with point-level supervision. It is the first to address this problem setup. Although point-level annotations are more informative than image-level annotations, they are almost as costly to acquire [10].

WISE-Net has two branches: (1) a localization network (L-Net) that predicts the location of each object; and (2) an embedding network (E-Net) that learns an embedding space where pixels of the same object are close. The segmentation masks for the located objects are obtained by grouping pixels with similar embeddings. At training time, while L-Net only requires point-level annotations, E-Net uses pseudo-labels generated by a class-agnostic object proposal method. We evaluate our approach on the PASCAL VOC [35], COCO [75], CityScapes [26], and KITTI [41] datasets.

The experiments show that our method (1) obtains competitive results compared to fully-supervised methods in certain scenarios; (2) outperforms fully- and weakly supervised methods with a fixed annotation budget; and (3) is a first strong baseline for instance segmentation with point-level supervision.

We have also proposed WISE-ILS, which is a Weakly Supervised Instance SEgmentation (WISE) method for Image-Level Supervision (ILS). In this case, the image label is whether an object class exists in the image. The label does not tell us how many objects are in the image nor their locations. Thus, the label is less informative than point-level annotations.

However, image-level labels carry a strong advantage over point-level labels. With minimal effort, we can obtain many images that have the objects of interest (like an image with a "car" present). For instance, obtaining such images can

be done using a scraping tool or an internet search engine, rather than having to manually label them.

Our framework, WISE-ILS, is significantly different from the two methods that exist for this setup [141, 24]. These two methods use classifiers that generate class-activation maps (CAMs) [109, 1] in order to identify peaks that represent specific locations of the object instances. At test time, the trained model obtains the object masks in two steps. First, it uses the gradient with respect to the input to get a rough mask for each object. Then, the rough mask is replaced with the best matching proposal masks, which are generated from a pretrained object proposal method [4, 93].

After training the CAM-based method (as in Zhou et al. [141], Cholakkal et al. [24]), WISE-ILS performs two steps. First, it generates masks for each object in the training set. Second, those masks are used to train a mask R-CNN [49], a state-of-the-art instance segmentation method. This simple procedure has achieved new state-of-the-art for this setup. Our results are based on evaluating our method on PASCAL VOC 2012, a standard dataset for weakly supervised methods, where we demonstrate major performance gains compared to existing methods with respect to mean average precision.

We summarize our contributions as follows. (1) We present a novel framework that can effectively train a fully supervised method on masks generated by training on image-level class labels; (2) we show that our framework is amenable to different localization and segmentation methods (for example, a density-based peak response map (PRM) [24] can be used for localization and RetinaMask [38] can be used for instance segmentation), and (3) we achieve new state-of-the-art results on a standard weakly supervised instance segmentation benchmark.

1.5 Crowd Localization in Dense Scenes

Crowd localization is the task of locating a large number of objects of interest in images. This differs from instance segmentation in that the model does not need to predict the masks of the objects. It just needs to predict their approximate location, which could be a form of bounding box, or point annotation that is roughly around the centers of the objects.

Methods that perform this task provide promising solutions for applications such as public safety, crowd monitoring, and traffic management. For congested scenes, some methods can only output the count [16]; however, for many applications, getting only the count is not enough. These applications demand localization as well, which could be critical for making decisions in high-risk environments like riots. Further, localization can help users understand the insights of the counting

method to judge whether it works in practice.

Crowd datasets often have point-level annotations. These annotations are necessary for many methods that learn to localize objects. Current methods use density-based and localization-based methods to locate objects in the image [68, 65, 71]. Unfortunately, given a large sequence of image frames from a fixed camera, it would require significant human effort to annotate the location of each object. Thus, we investigate weaker supervision where only the image-level count is provided for each training image.

At first glance, it would seem that the effort required for collecting image-level counts and point-level annotations are equivalent. After all, a human annotator needs to point to every object in the image to count them. But this is not true for several cases.

First, in the case where training images contain 5 or fewer objects, the annotator can obtain the object count much faster than with point annotations [16]. Second, in the case of having a large sequence of image frames, it is much easier to count the number of objects across these frames than to locate each object. The annotator can first count each object in the first frame, which roughly takes the same amount as collecting point-level annotations. But if the number of objects remains similar in the subsequent frames, the annotator would spend less time acquiring the counts for those frames. Third, in the case of registration-based systems, object counts can come for free. For instance, if a manager ordered a certain number of cans to be put in a glass-windowed fridge, then the images taken of that fridge are already labeled with the count of cans. Therefore, many cases exist where acquiring image counts can take much less time than point-level annotations.

The few works that exist for localization based on counting are limited to datasets with few objects [119, 40]. They treat the problem as classification and use CAM-based methods [109, 1] to identify regions where the objects are as a localization heatmap. They also leverage proposals [93] which they rank based on the localization heatmap. At test time, the proposals with the highest scores are considered the predicted objects of interest. While this framework is a standard for this setup, it is limited to datasets where the number of objects of interest is usually one, as in the PASCAL VOC dataset.

On the other hand, one work exists that detects multiple objects in an image using count-supervision [40]. Known as C-WSL, it trains by performing two steps every iteration. In the first step, it selects non-overlapping proposals with the highest probability of being the ground-truth, using the provided count information. In the second step, a classification network is updated to maximize its probabilities in predicting those selected proposals. Thus, it is a block coordinate optimization paradigm. This procedure converges when the selected proposals do not change across iterations. Afterwards, a faster region-based convolutonal neural networks

(R-CNN) [42] is trained on the selected proposals generated for each training example. The trained faster R-CNN is then used to detect the objects on the test set.

However, a limitation of this framework is it uses a CAM-based method to score the proposals. As a result, it cannot be directly applied to crowd datasets where the same object class exists in every image. Further, the goal of our setup is to locate the objects in addition to detecting them. As discussed in the counting setup, having a method that learns to perform the more difficult task of detection can yield worse localization results.

Nevertheless, we adapt C-WSL [40] to our setup by proposing a weakly supervised localization method that we call as WSLM. For this method, we generate proposals in the image and rank them based on those that are most likely to be the objects of interest. This procedure is heavily influenced by the expectation-maximization paradigm [30] used in many multiple instance learning [33] setups.

Different from C-WSL, WSLM converts the selected proposals into point-level annotations, and then updates an LCFCN (our counting method) on these point-level annotations. Hence, we leverage the localization and counting abilities of LCFCN to localize and count objects. At test time, the trained LCFCN is used directly to localize the objects of interest.

Since no direct relevant work exists for this particular setup, we compared our method against several baselines, namely Glance [16] and WSLM with different proposal ranking methods. We benchmarked our method against standard crowd counting datasets. WSLM achieved better results than the baselines with respect to mean-absolute error, a counting metric, and grid mean absolute error [45], a localization metric.

We summarize our contributions as follows: (1) we show a novel framework that can locate objects from count-level supervision only for crowd datasets, and (2) we show that state-of-the-art results can be achieved by a novel modification of several baseline frameworks.

1.6 Summary of Contributions

Our work contains novel methods where models don't need fully annotated labels to learn to perform their task. With these methods, we achieved state-of the art-results in four different setups. We display a summary of these setups, and present the novelty of the methods in Table 1.1.

Method	Task	Supervision	Overview
LCFCN	counting	point level	Trains a segmentation network using a novel loss function. This loss encourages a single blob per object. The predicted count is the number of blobs in the image (Figure 1.5).
WISE-Net	ins. segm.	point level	A trained LCFCN finds the objects in the image. Then, a trained embedding network groups pixels to get the mask for each of these objects.
WISE-ILS	ins. segm.	image label	A trained PRM [141] and a proposal network are used to get object masks for each training image as pseudo labels. Then, a mask R-CNN is trained on those pseudo labels.
WSLM	localization	object count	A trained Glance method outputs a regression activation map to get a heatmap of the object locations. Using an RPN [49], a set of proposals are generated for the training images, which are scored based on that heatmap. The best proposals are selected using CSR [40] and their centroids are chosen as point annotations, considered as pseudo labels. Then, an LCFCN is trained on those pseudo labels.

Table 1.1: Summary of the proposed methods and their respective setups.

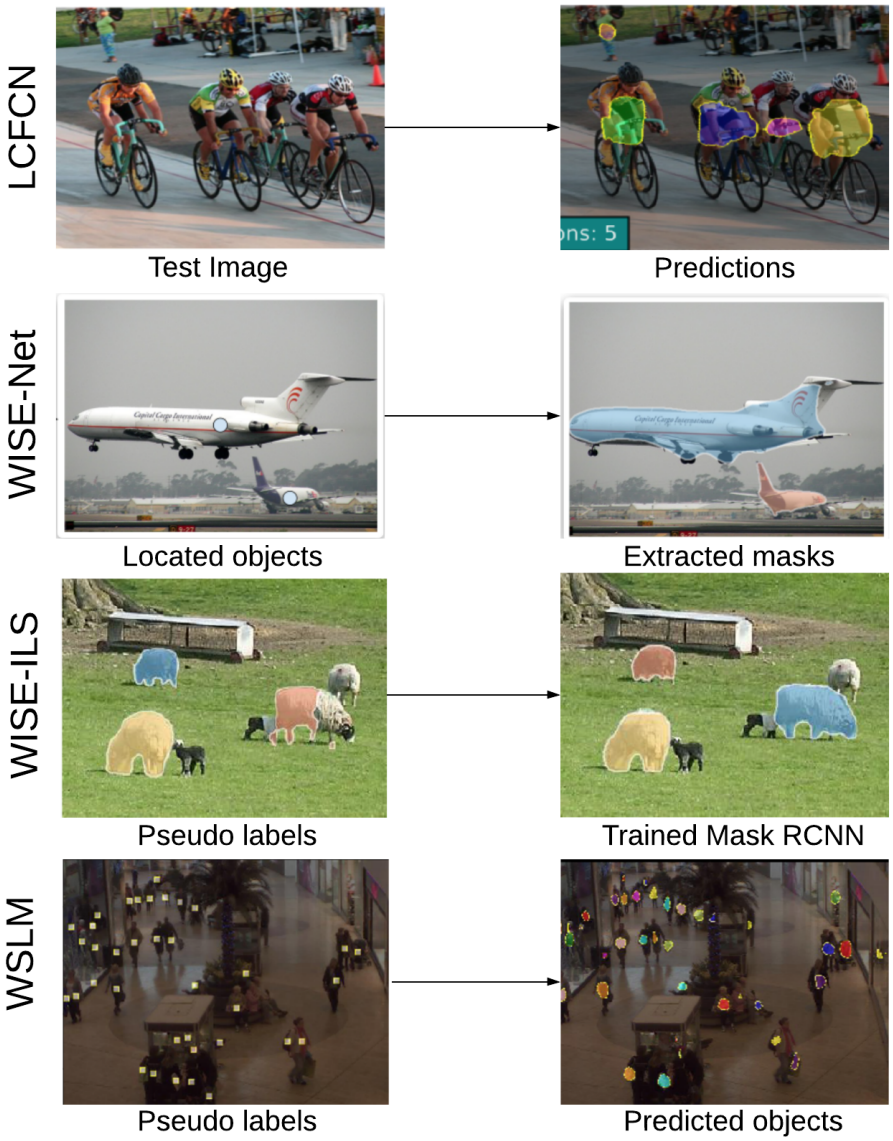


Figure 1.5: A summary of our proposed methods. LCFCN predicts blobs for a test image to obtain the count. WISE-Net first locates the objects using an LCFCN and then extracts their masks using an embedding network. WISE-ILS first obtains pseudo labels with the help of a proposal network. A mask R-CNN train on those pseudo labels to learn to perform instance segmentation. WSLM first obtains point annotations as pseudo labels and then trains an LCFCN to perform localization.

Chapter 2

Where are the Blobs

Object counting is an important task in computer vision due to its growing demand in applications such as surveillance, traffic monitoring, and counting everyday objects. State-of-the-art methods use regression-based optimization where they explicitly learn to count the objects of interest. These often perform better than detection-based methods that need to learn the more difficult task of predicting the location, size, and shape of each object. However, we propose a detection-based method that does not need to estimate the size and shape of the objects and that outperforms regression-based methods. Our contributions are three-fold: (1) we propose a novel loss function that encourages the network to output a single blob per object instance using point-level annotations only; (2) we design two methods for splitting large predicted blobs between object instances; and (3) we show that our method achieves new state-of-the-art results on several challenging datasets including the Pascal VOC and the Penguins dataset. Our method even outperforms those that use stronger supervision such as depth features, multi-point annotations, and bounding-box labels.

2.1 Introduction

Object counting is an important task in computer vision with many applications in surveillance systems [128, 132], traffic monitoring [29, 45], ecological surveys [8], and cell counting [25, 68]. In traffic monitoring, counting methods can be used to track the number of moving cars, pedestrians, and parked cars. They can also be used to monitor the count of different species such as penguins, which is important for animal conservation. Furthermore, counting methods have been used for counting objects present in everyday scenes where objects of interest come from a large number of classes such as the Pascal VOC dataset [35].

Many challenges are associated with object counting. Models need to learn the variability of the objects in terms of shape, size, pose, and appearance. Moreover, objects may appear at different angles and resolutions, and may be partially occluded (see Fig. 2.1). Also, the background, weather conditions, and illuminations can vary widely across the scenes. Therefore, the model needs to be robust enough

Figure 2.1: Qualitative results on the Penguins [8] and PASCAL VOC datasets [35]. Our method explicitly learns to localize object instances using only point-level annotations. The trained model then outputs blobs where each unique color represents a predicted object of interest. Note that the predicted count is simply the number of predicted blobs.

to recognize objects in the presence of these variations in order to perform efficient object counting.

Due to these challenges, regression-based models such as "Glance" and object density estimators have consistently defined state-of-the-art results in object counting [16, 89]. This is because their loss functions are directly optimized for predicting the object count. In contrast, detection-based methods need to optimize for the more difficult task of estimating the location, shape, and size of the object instances. Indeed, perfect detection implies perfect count as the count is simply the number of detected objects. However, models that learn to detect objects often lead to worse results for object counting [16]. For this reason, we look at an easier task than detection by focusing on the task of simply localizing object instances in the scene. Predicting the exact size and shape of the object instances is not necessary and usually poses a much more difficult optimization problem. Therefore, we propose a novel loss function that encourages the model to output instance regions such that each region contains a single object instance (i.e. a single point-level annotation). Similar to detection, the predicted count is the number of predicted instance regions (see Fig. 2.1). Our model only requires point supervision which is a weaker supervision than bounding-box, and per-pixel annotations used by most detection-based methods [103, 100, 9]. Consequently, we can train our model for most counting datasets as they often have point-level annotations.

This type of annotation is cheap to acquire as it requires lower human effort than bounding box and per-pixel annotations [10]. Point-level annotations provide a rough estimate of the object locations, but not their sizes nor shapes. Our counting method uses the provided point annotations to guide its attention to the object instances in the scenes in order to learn to localize them. As a result, our model has the flexibility to predict different sized regions for different object instances, which makes it suitable for counting objects that vary in size and shape. In contrast,

state-of-the-art density-based estimators often assume a fixed object size (defined by the Gaussian kernel) or a constrained environment [45] which makes it difficult to count objects with different sizes and shapes.

Given only point-level annotations, our model uses a novel loss function that (i) allows it to predict the semantic segmentation labels for each pixel in the image (similar to [10]) and (ii) encourages it to output a segmentation blob for each object instance. During the training phase, the model learns to split the blobs that contain more than one point annotation and to remove the blobs that contain no point-level annotations.

Our experiments show that our method achieves superior object counting results compared to state-of-the-art counting methods including those that use stronger supervision such as per-pixel labels. Our benchmark uses datasets representing different settings for object counting: Mall [17], UCSD [14], and ShanghaiTech B [136] as crowd datasets; MIT Traffic [129] and Park lot [29] as surveillance datasets; Trancos [45] as a traffic monitoring dataset; and Penguins [8] as a population monitoring dataset. We also show counting results for the PASCAL VOC [35] dataset which consists of objects present in natural, "everyday" images. We also study the effect of using different parts of the proposed loss function against counting and localization performance.

We summarize our contributions as follows: (1) we propose a novel loss function that encourages the network to output a single blob per object instance using point-level annotations only; (2) we design two methods for splitting large predicted blobs between object instances; and (3) we show that our method achieves new state-of-the-art results on several challenging datasets including the Pascal VOC and the Penguins dataset.

This chapter is organized as follows: Section 2.2 presents related works on object counting. Section 2.3 describes the proposed approach, and Section 2.4 describes our experiments and results. Finally, we present the conclusion in Section 2.6.

2.2 Related Work

Object counting has received significant attention over the past years [97, 16, 68]. It can be roughly divided into three categories [81]: (1) counting by clustering, (2) counting by regression, and (3) counting by detection.

Early work in object counting use *clustering-based methods*. These are unsupervised approaches where objects are clustered based on features such as appearance and motion cues [97, 121]. Rabaud and Belongie [97] proposed using feature points which are detected by motion and appearance cues and are tracked through

time using PCA [111]. The objects are then clustered based on similar features. Tu et al. [121] used an expectation-maximization method that cluster individuals in crowds based on head and shoulder features. These methods use basic features and often perform poorly for counting compared to deep learning approaches. Another drawback is that these methods only work for video sequences, rather than still images.

Counting by regression methods have defined state-of-the-art results in many benchmarks. They were shown to be faster and more accurate than other groups such as counting by detection. These methods include Glance and density-based methods that explicitly learn how to count rather than optimize for a localization-based objective. Lempitsky and Zisserman [68] proposed the first method that used object density to count people. They transform the point-level annotation matrix into a density map using a Gaussian kernel. Then, they train their model using a least-squares objective to predict the density map. One major challenge is determining the optimal size of the Gaussian kernel which highly depends on the object sizes. As a result, Zhang et al. [136] proposed a deep learning method that adjusted the kernel size using a perspective map. This assumes fixed camera images such as those used in surveillance applications. Onoro-Rubio Onoro-Rubio and López-Sastre [89] extended this method by proposing a perspective-free multi-scale deep learning approach. However, this method cannot be used for counting everyday objects as their sizes vary widely across the scenes as it is highly sensitive to the kernel size.

A straight-forward method for counting by regression is "Glance" [16], which explicitly learns to count using image-level labels only. Glance methods are efficient if the object count is small [16]. Consequently, the authors proposed a grid-based counting method, denoted as "subitizing", in order to count a large number of objects in the image. This method uses Glance to count objects at different non-overlapping regions of the image, independently. While Glance is easy to train and only requires image-level annotation, the "subitizing" method requires a more complicated training procedure that needs full per-pixel annotation ground-truth.

Counting by detection methods first detect the objects of interest and then simply count the number of instances. Successful object detection methods rely on bounding boxes [103, 100, 78] and per-pixel labels [79, 53, 139] ground-truth. Perfect object detection implies perfect counting. However, Chattopadhyay et al. [16] showed that Fast RCNN [42], a state-of-the-art object detection method, performs worse than Glance and subitizing-based methods. This is because the detection task is challenging in that the model needs to learn the location, size, and shape of object instances that are possibly heavily occluded. While several works [16, 89, 68] suggest that counting by detection is infeasible for surveillance scenes where objects are often occluded, we show that learning a notion of localization can help

the model improve counting.

Similar to our method is the line of work proposed by Arteta et al. [5, 6, 7]. They proposed a method that detects overlapping instances based on optimizing a tree-structured discrete graphical model. While their method showed promising detection results using point-level annotations only, it performed worse for counting than regression-based methods such as [68].

Our method is also similar to segmentation methods such as U-net [105] which learns to segment objects using a fully-convolutional neural network. Unlike our method, U-net requires the full per-pixel instance segmentation labels, whereas we use point-level annotations only.

2.3 Localization-based Counting FCN

Our model is based on the fully convolutional neural network (FCN) proposed by Long et al. [79]. We extend their semantic segmentation loss to perform object counting and localization with point supervision. We denote the novel loss function as *localization-based counting loss* (LC) and, we refer to the proposed model as LC-FCN. Next, we describe the proposed loss function, the architecture of our model, and the prediction procedure.

2.3.1 The Proposed Loss Function

LC-FCN uses a novel loss function that consists of four distinct terms. The first two terms, the image-level and the point-level loss, encourage the model to predict the semantic segmentation labels for each pixel in the image. This is based on the weakly supervised semantic segmentation algorithm proposed by Bearman et al. [10]. These two terms alone are not suitable for object counting as the predicted blobs often group many object instances together (see the ablation studies in Section 2.4). The last two terms encourage the model to output a unique blob for each object instance and remove blobs that have no object instances. Overall, the four loss terms is equivalent to applying the cross-entropy loss on strategically placed foreground and background pixels. Note that LC-FCN only requires point-level annotations that indicate the locations of the objects rather than their sizes, and shapes.

Let T represent the point annotation ground-truth matrix which has label c at the location of each object (where c is the object class) and zero elsewhere. Our model uses a *softmax* function to output a matrix S where each entry S_{ic} is the probability that pixel i belongs to category c. The proposed loss function can be

written as:

$$\mathcal{L}(S,T) = \underbrace{\mathcal{L}_I(S,T)}_{\text{Image-level loss}} + \underbrace{\mathcal{L}_P(S,T)}_{\text{Point-level loss}} + \underbrace{\mathcal{L}_S(S,T)}_{\text{Split-level loss}} + \underbrace{\mathcal{L}_F(S,T)}_{\text{False positive loss}} , \quad (2.1)$$

which we describe in detail next.

Image-level loss.

The goal of this loss term is to predict at least one pixel for object classes belonging to the image, and no pixels for the classes that do not belong to the image. It performs a global max-pooling operation on the output per-pixel probabilities for an image and applies a cross-entropy loss against the class labels of that image. Let C_e be the set of classes present in the image. For each class $c \in C_e$, $\mathcal{L}_I$ increases the probability that the model labels at least one pixel as class c. Also, let $C_{\neg e}$ be the set of classes not present in the image. For each class $c \in C_{\neg e}$, the loss decreases the probability that the model labels any pixel as class c. C_e and $C_{\neg e}$ can be obtained from the provided ground-truth point-level annotations. More formally, the image level loss is computed as follows:

$$\mathcal{L}_I(S,T) = -\frac{1}{|C_e|}\sum_{c\in C_e}\log(S_{t_c c}) - \frac{1}{|C_{\neg e}|}\sum_{c\in C_{\neg e}}\log(1-S_{t_c c}) , \quad (2.2)$$

where $t_c = \text{argmax}_{i\in\mathcal{I}} S_{ic}$. For each category present in the image, at least one pixel should be labeled as that class. For classes that do not exist in the image, none of the pixels should belong to that class. Note that we assume that each image has at least one background pixel; therefore, the background class belongs to C_e.

Point-level loss.

This term encourages the model to correctly label the small set of supervised pixels $\mathcal{I}_s$ contained in the ground-truth. It uses the standard cross-entropy loss between the per-pixel probability output and the provided point-level annotations. $\mathcal{I}_s$ represents the locations of the object instances. This is formally defined as,

$$\mathcal{L}_P(S,T) = -\sum_{i\in\mathcal{I}_s}\log(S_{iT_i}) , \quad (2.3)$$

where T_i represents the true label of pixel i. Note that this loss ignores all the pixels that are not annotated.

Split-level loss.

$\mathcal{L}_S$ discourages the model from predicting blobs that have two or more point-annotations. Therefore, if a blob has n point annotations, this loss enforces it to be split into n blobs, each corresponding to a unique object. This loss function is optimized as follows. The splits are made by first finding boundaries between object pairs. Those boundaries are set as background pixels where the cross-entropy loss is computed between the per-pixel output probabilities and the background pixels . As a result, The model outputs a binary matrix $\mathcal{F}$ where pixel i is foreground if $\text{argmax}_k S_{ik} > 0$, and background, otherwise.

We apply the connected components algorithm proposed by Wu et al. [131] to find the blobs B in the foreground mask $\mathcal{F}$. We only consider the blobs with two or more ground truth point annotations $\bar{B}$. We propose two methods for splitting blobs (see Fig. 2.2),

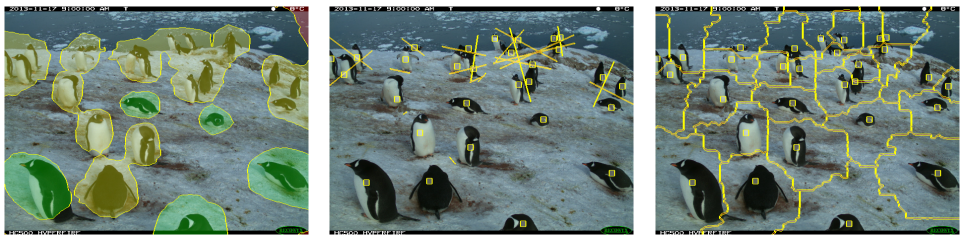
1. *Line split method*. The goal of this method is to find a boundary that allows us to split a blob with more than one object. If two points are within a single blob, then we first draw a straight line between those two points. Then we look into every possible line that is perpendicular to that line. The line with pixels that have the highest average background probability is selected. Then, the cross-entropy loss is computed between each of these pixels and the per-pixel output probabilities of the model.

   More formally, for each blob b in $\bar{B}$ we pair each point with its closest point resulting in a set of pairs b_P. For each pair $(p_i, p_j) \in b_P$ we use a scoring function to determine the best segment E that is perpendicular to the line between p_i and p_j. The segment lines are within the predicted blob and they intersect the blob boundaries. The scoring function $z(\cdot)$ for segment E is computed as,

   $$z(E) = \frac{1}{|E|} \sum_{i \in E} S_{i0} \ , \tag{2.4}$$

   which is the mean of the background probabilities belonging to segment E (where 0 is the background class). The best edge E_{best} is defined as the set of pixels representing the edge with the highest probability of being background among all the perpendicular lines. This determines the "most likely" edge of separation between the two objects. Then we set T_b as the set of pixels representing the best edges generated by the line split method.

2. *Watershed split method*. This consists of global and local segmentation procedures. For the global segmentation, we apply the watershed segmentation algorithm [11] globally on the input image where we set the ground-truth



Predicted blobs | Line splits | Watershed splits

Figure 2.2: **Split methods.** Comparison between the line split, and the watershed split. The loss function identifies the boundary splits (shown as yellow lines). Yellow blobs represent those with more than one object instance, and red blobs represent those that have no object instance. Green blobs are true positives. The squares represent the ground-truth point annotations.

point-annotations as the seeds. The segmentation is applied on the distance transform of the foreground probabilities, which results in k segments where k is the number of point-annotations in the image.

For the local segmentation procedure, we apply the watershed segmentation only within each blob b in $\bar{B}$ where we use the point-annotation ground-truth inside them as seeds. This adds more importance to splitting big blobs when computing the loss function. Finally, we define T_b as the set of pixels representing the boundaries determined by the local and global segmentation.

Fig. 2.2 shows the split boundaries using the line split and the watershed split methods (as yellow lines). Given T_b, we compute the split loss as follows,

$$\mathcal{L}_S(S, T) = -\sum_{i \in T_b} \alpha_i \log(S_{i0}), \tag{2.5}$$

where S_{i0} is the probability that pixel i belongs to the background class and α_i is the number of point-annotations in the blob in which pixel i lies. This encourages the model to focus on splitting blobs that have the most point-level annotations. The intuition behind this method is that learning to predict a boundary between the object instances allows the model to distinguish between them. As a result, the penalty term encourages the model to output a single blob per object instance.

We emphasize that it is not necessary to get the right edges in order to accurately count. It is only necessary to make sure we have a positive region on each object and a negative region between objects. Other heuristics are possible to construct a negative region which could still be used in our framework. For example,

fast label propagation methods proposed in Nutini et al. [88, 87] can be used to determine the boundaries between the objects in the image. Note that these 4 loss functions are only used during training. The framework does not split or remove false positive blobs at test time. The predictions are based purely on the blobs obtained from the probability matrix S.

False Positive loss.

The goal of this loss term is to prevent the model from predicting blobs that have no objects in them. For a given training image, the model outputs blobs using the procedure described in the split-level loss section. The pixels belonging to the blobs without objects define background pixels. Then the cross-entropy loss is computed between the model's per-pixel probabilities and these background pixels. More formally, the loss function is defined as

$$\mathcal{L}_F(S, T) = -\sum_{i \in B_{fp}} \log(S_{i0}), \tag{2.6}$$

where B_{fp} is the set of pixels constituting the blobs predicted for each class (except the background class) that contain no ground-truth point annotations (note that S_{i0} is the probability that pixel i belongs to the background class). All the predictions within B_{fp} are considered false positives (see the red blobs in Fig. 2.5). Therefore, optimizing this loss term results in less false positive predictions as shown in the qualitative results in Fig. 2.5. The experiments show that this loss term is extremely important for accurate object counting.

2.3.2 LC-FCN Architecture and Inference

LC-FCN can be any FCN architecture such as FCN8 architecture [79], Deeplab [19], Tiramisu [53], and PSPnet [139]. LC-FCN consists of a backbone that extracts the image features. The backbone is an Imagenet pretrained network such as VGG16 or ResNet-50 [113, 31]. The image features are then upscaled using an upsampling path to output a score for each pixel i indicating the probability that it belongs to class c (see Fig. 2.3).

We predict the number of objects for class c through the following three steps: (i) the upsampling path outputs a matrix Z where each entry Z_{ic} is the probability that pixel i belongs to class c; then (ii) we generate a binary mask F, where pixel $F_{ic} = 1$ if $\arg\max_k Z_{ik} = c$, and 0 otherwise; lastly (iii) we apply the connected components algorithm [131] on F to get the blobs for each class c. The count is the number of predicted blobs (see Fig. 2.3).

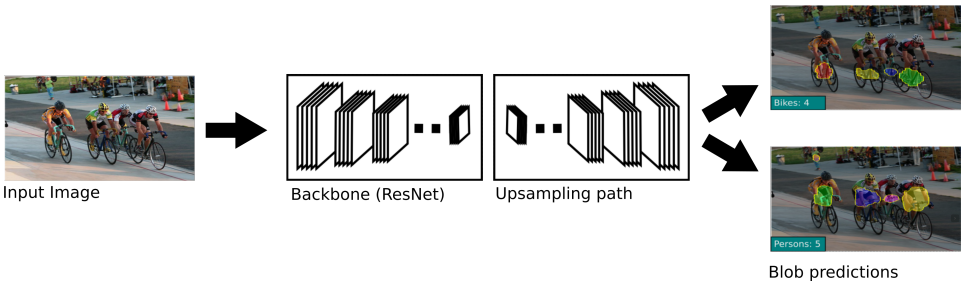


Figure 2.3: Given an input image, our model first extracts features using a backbone architecture such as ResNet. The extracted features are then upsampled through the upsampling path to obtain blobs for the objects. In this example, the model predicts the blobs for persons and bikes for an image in the PASCAL VOC 2007 dataset.

	Separated		**Mixed**	
Method	Max	Median	Max	Median
Density-only [8]	8.11	5.01	9.81	7.09
With seg. and depth [8]	6.38	3.99	5.74	3.42
With seg and no depth [8]	5.77	3.41	5.35	3.26
Marsde *et al.* [83]	5.8	x	x	x
Glance	6.08	5.49	1.84	2.14
LC-FCN8	**3.74**	**3.28**	1.62	1.80
LC-ResFCN	3.96	3.43	**1.50**	**1.69**

Table 2.1: **Penguins datasets.** Evaluation of our method against previous state-of-the-art methods. The evaluation is made across the four setups explained in the dataset description. Note that the methods used by Arteta et al. [8] use annotations from multiple labelers and depth.

2.4 Experiments

In this section we describe the evaluation metrics, the training procedure, and present the experimental results and discussion.

2.4.1 Setup

Evaluation Metric.

For datasets with single-class objects, we report the mean absolute error (MAE) which measures the deviation of the predicted count p_i from the true count c_i, computed as $\frac{1}{N}\sum_i |p_i - c_i|$. MAE is a commonly used metric for evaluating object counting methods [15, 115]. For datasets with multi-class objects, we report the mean root mean square error (mRMSE) as used in Chattopadhyay et al. [16] for the PASCAL VOC 2007 dataset. We measure the localization performance using the average mean absolute error (GAME) as in Guerrero et al. [45]. Since our model predicts blobs instead of a density map, GAME might not be an accurate localization measure. Therefore, in section 2.4.3 we use the F-Score metric to assess the localization performance of the predicted blobs against the point-level annotation ground-truth.

Training Procedure.

We use the Adam [58] optimizer with a learning rate of 10^{-5} and weight decay of 5×10^{-5}. We use the provided validation set for early stopping only. During training, the model uses a batch size of 1 which can be an image of any size. We double our training set by applying the horizontal flip augmentation method on each image. Finally, we report the prediction results on the test set. We compare between three architectures: FCN8 [79]; ResFCN which is FCN8 that uses ResNet-50 as the backbone instead of VGG16; and PSPNet [139] with ResNet-101 as the backbone. We use the watershed split procedure in all our experiments.

2.4.2 Results and Discussion

Penguins Dataset [8].

The Penguins dataset comprises images of penguin colonies located in Antarctica. We use the two dataset splits as in Arteta et al. [8]. In the "separated" dataset split, the images in the training set come from different cameras than those in the test set. In the "mixed" dataset split, the images in the training set come from the same cameras as those in the test set. In Table 2.1, the MAE is computed with respect to the Max and Median count (as there are multiple annotators). Our methods significantly outperform the methods proposed by Arteta et al. [8] in all of the four settings, although their methods use depth features and the multiple annotations provided for each penguin. This suggests that LC-FCN can learn to distinguish between individual penguins despite the heavy occlusions and crowding.

Method	MAE	GAME(1)	GAME(2)	GAME(3)
Lemptisky+SIFT [45]	13.76	16.72	20.72	24.36
Hydra CCNN [89]	10.99	13.75	16.69	19.32
FCN-MT [135]	5.31	-	-	-
FCN-HA [134]	4.21	-	-	-
CSRNet [71]	3.56	5.49	8.57	15.04
Glance	7.0	-	-	-
LC-FCN8	4.53	7.00	10.66	16.05
LC-ResFCN	**3.39**	5.2	7.92	12.57
LC-PSPNET	3.57	**4.98**	**7.42**	**11.67**

Table 2.2: **Trancos dataset.** Evaluation of our method against previous state-of-the-art methods, comparing the mean absolute error (MAE) and the grid average mean absolute error (GAME) as described in Guerrero et al. [45].

Trancos Dataset [89].

The Trancos dataset comprises images taken from traffic surveillance cameras located along different roads. The task is to count the vehicles present in the regions of interest of the traffic scenes. Each vehicle is labeled with a single point annotation that represents its location in the image. We observe in Table 2.2 that our method achieves new state-of-the-art results for counting and localization. Note that $GAME(L)$ subdivides the image using a grid of 4^L non-overlapping regions, and the error is computed as the sum of the mean absolute errors in each of these subregions. For our method, the predicted count of a region is the number of predicted blob centers in that region. This provides a rough assessment of the localization performance. Compared to the methods in Table 2.2, LC-FCN does not require a perspective map nor a multi-scale approach to learn objects of different sizes. These results suggest that LC-FCN can accurately localize and count extremely overlapping vehicles.

Parking Lot [29].

The dataset comprises surveillance images taken at a parking lot in Curitiba, Brazil. We used the PUCPR subset of the dataset where the first 50% of the images was set as the training set and the last 50% as the test set. The last 20% of the training set was set as the validation set for early stopping. The ground truth consists of a

Method	mRMSE	mRMSE-nz	m-relRMSE	m-relRMSE-nz
Glance-noft-2L [16]	0.50	1.83	0.27	0.73
Aso-sub-ft-3 $\times$ 3 [16]	0.42	1.65	0.21	0.68
Faster-RCNN [16]	0.50	1.92	0.26	0.85
LC-ResFCN	**0.31**	**1.20**	**0.17**	**0.61**
LC-PSPNet	0.35	1.32	0.20	0.70

Table 2.3: **PASCAL VOC.** We compare against the methods proposed in [16]. Our model evaluates on the full test set, whereas the other methods take the mean of ten random samples of the test set evaluation. Note that Aso-sub-ft-3 uses full per-pixel supervision

bounding box for each parked car since this dataset is primarily used for the detection task. Therefore, we convert them into point-level annotations by taking the center of each bounding box. Table 2.5 shows that LC-FCN significantly outperforms Glance in MAE. LC-FCN8 achieves only 0.21 average miscount per image although many images contain more than 20 parked cars. This suggests that explicitly learning to localize parked cars can perform better in counting than methods that explicitly learn to count from image-level labels (see Fig. 2.5 for qualitative results). Note that this is the first counting method being applied on this dataset.

MIT Traffic [128].

This dataset consists of surveillance videos taken from a single fixed camera. It has 20 videos, which are split into a training set (Videos 1-8), a validation set (Videos 0-10), and a test set (Videos 11-20). Each video frame is provided with a bounding box indicating each pedestrian. We convert them into point-level annotations by taking the center of each bounding box. Table 2.5 shows that our method significantly outperforms Glance, suggesting that learning a localization-based objective allows the model to ignore the background regions that do not contribute to the object count. As a result, LC-FCN is less likely to overfit on irrelevant features from the background. To the best of our knowledge, this is the first counting method being applied on this dataset.

Pascal VOC 2007 [35].

We use the standard training, validation, and test split as specified in Everingham et al. [35]. We use the point-level annotation ground-truth provided by Bearman

Methods	UCSD	Mall	ShanghaiTech B
FCN-rLSTM [134]	1.54	-	-
MoCNN [63]	-	2.75	-
CNN-boosting [125]	1.10	2.01	-
M-CNN [136]	1.07	-	26.4
CP-CNN [114]	-	-	20.1
CSRNet [71]	1.16	-	**10.6**
LC-FCN8	1.51	2.42	13.64
LC-ResFCN	**0.99**	2.12	25.89
LC-PSPNet	1.01	**2.00**	21.61

Table 2.4: Crowd datasets MAE results.

Figure 2.4: Predicted blobs on a ShanghaiTech B test image.

et al. [10] to train our LC-FCN methods. We evaluated against the count of the non-difficult instances of the Pascal VOC 2007 test set.

Table 2.3 compares the performance of LC-FCN with different methods proposed by Chattopadhyay et al. [16]. We point the reader to Chattopadhyay et al. [16] for a description of the evaluation metrics used in the table. We show that LC-FCN achieves new state-of-the-art results with respect to mRMSE. We see that LC-FCN outperforms methods that explicitly learn to count although learning to localize objects of this dataset is a very challenging task. Further, LC-FCN uses weaker supervision than Aso-sub and Seq-sub as they require the full per-pixel labels to estimate the object count for different image regions.

	MIT Traffic		PKLot		Trancos		Penguins Separated	
Method	**MAE**	**FS**	**MAE**	**FS**	**MAE**	**FS**	**MAE**	**FS**
Glance	1.57	-	1.92	-	7.01	-	6.09	-
$\mathcal{L}_I + \mathcal{L}_P$	3.11	0.38	39.62	0.04	38.56	0.05	9.81	0.08
$\mathcal{L}_I + \mathcal{L}_P + \mathcal{L}_S$	1.62	0.76	9.06	0.83	6.76	0.56	4.92	0.53
$\mathcal{L}_I + \mathcal{L}_P + \mathcal{L}_F$	1.84	0.69	39.60	0.04	38.26	0.05	7.28	0.04
LC-ResFCN	1.26	**0.81**	10.16	0.84	**3.39**	0.68	3.96	0.63
LC-FCN8	**0.91**	0.69	**0.21**	**0.99**	4.53	**0.54**	**3.74**	**0.61**

Table 2.5: **Quantitative results.** Comparison of different parts of the proposed loss function for counting and localization performance.

Crowd Counting Datasets.

Table 2.4 reports the MAE score of our method on 3 crowd datasets using the setup described in the survey paper [115]. For this experiment, we show our results using ResFCN as the backbone with the Watershed split method. We see that our method achieves competitive performance for crowd counting. Fig. 2.4 shows the predicted blobs of our model on a test image of the ShanghaiTech B dataset. We see that our model predicts a blob on the face of each individual. This is expected since the ground-truth point-level annotations are marked on each person's face.

2.4.3 Ablation Studies

Localization Benchmark.

Since robust localization is useful in many computer vision applications, we use the F-Score measure to directly assess the localization performance of our model. F-Score is a standard measure for detection as it considers both precision and recall, F-Score $= \frac{2\text{TP}}{2\text{TP}+\text{FP}+\text{FN}}$, where the number of true positives (TP) is the number of blobs that contain at least one point annotation; the number of false positives (FP) is the number of blobs that contain no point annotation; and the number of false negatives (FN) is the number of point annotations minus the number of true positives. Table 2.5 shows the localization results of our method on several datasets.

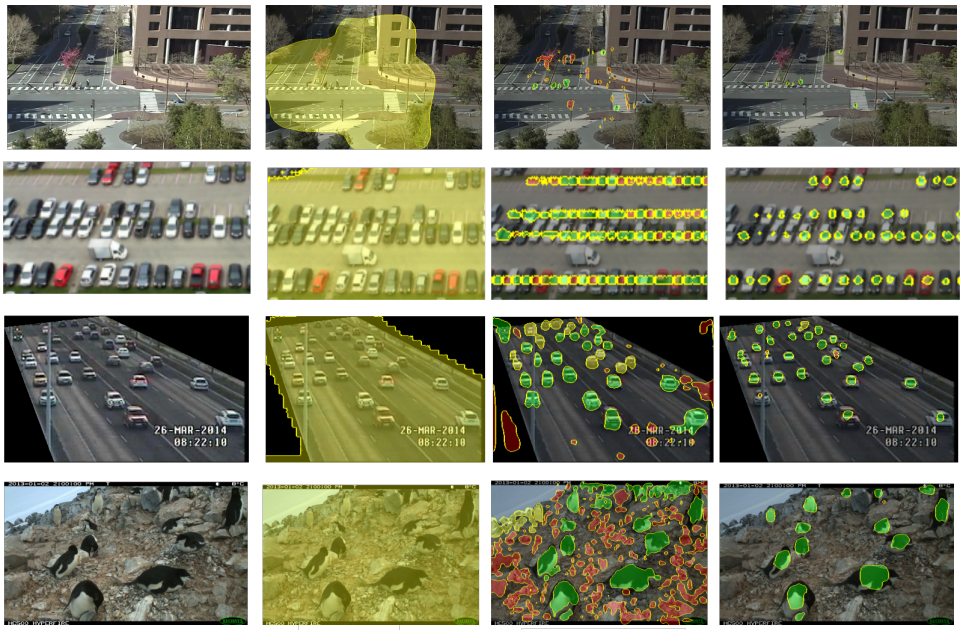


Figure 2.5: Qualitative results of LC-FCN trained with different terms of the proposed loss function. (a) Test images obtained from MIT Traffic, Parking Lot, Trancos, and Penguins. (b) Prediction results using only image-level and point-level loss terms. (c) Prediction results using image-level, point-level, and split-level loss terms. (d) Prediction results trained with the full proposed loss function. The green blobs and red blobs indicate true positive and false positive predictions, respectively. Yellow blobs represent those that contain more than one object instance.

Loss Function Analysis.

We assess the effect of each term of the loss function on counting and localization results. We start by looking at the results of a model trained with the image-level loss $\mathcal{L}_I$ and the point-level loss $\mathcal{L}_P$ only. These two terms were used for semantic segmentation using point annotations [10]. We observe in Fig. 2.5(b) that a model using these two terms results in a single blob that groups many object instances together. Consequently, this performs poorly in terms of the mean absolute error and the F-Score (see Table 2.5). As a result, we introduced the split-level loss function $\mathcal{L}_S$ that encourages the model to predict blobs that do not contain more than one point-annotation. We see in Fig. 2.5(c) that a model using this additional loss term predicts several blobs as object instances rather than one large single blob.

Split method	Trancos	Penguins
LC-ResFCN (L)	4.77	1.89
LC-ResFCN (W)	3.34	0.95

Figure 2.6: **Split Heuristics Analysis.** Comparison between the watershed split method and the line split method against the validation MAE score.

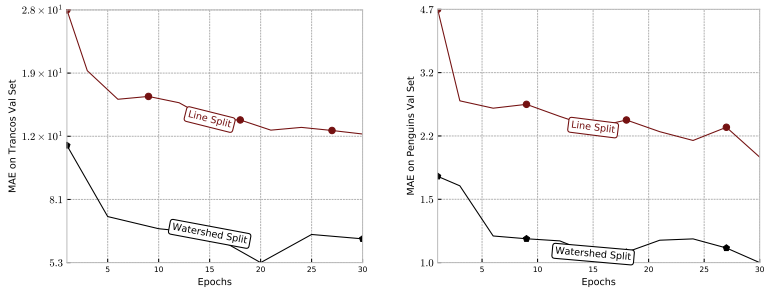


Figure 2.7: **Split Heuristics Analysis.** Comparison between the watershed split method and the line split method against the validation MAE score.

However, since $\mathcal{L}_I + \mathcal{L}_P + \mathcal{L}_S$ does not penalize the model for predicting blobs with no point annotations, it can often lead to many false positives. Therefore, we introduce the false positive loss $\mathcal{L}_F$ that discourages the model from predicting blobs with no point annotations. By adding this loss term to the optimization, LC-FCN achieves significant improvement as seen in the qualitative and quantitative results (see Fig. 2.5(d) and Table 2.5). Further, including only the split-level loss leads to predicting a huge number of small blobs, leading to many false positives which makes performance worse. Combining it with the false-positive loss avoids this issue which leads to a net improvement in performance. On the other hand, using only the false positive loss it tends to predict one huge blob.

Split Heuristics Analysis.

In Fig. 2.4.3 we show that the watershed split achieves better MAE on Trancos and Penguins validation sets. Further, using the watershed split achieves much faster improvement on the validation set with respect to the number of epochs. This suggests that using proper heuristics to identify the negative regions is important, which leaves an open area for future work.

2.5 Limitations

LCFCN might be sensitive to the threshold used to get the blobs. For instance, if the threshold is set to 0.5, and the probability output for multiple objects, say 10, is 0.25, then LCFCN will output a count of zero. In this case it is likely that there is at least one object in the image, which LCFCN might fail to count.

LCFCN's loss function can be highly unstable in the case when many objects overlap. The optimization might oscillate between satisfying the split-level loss and the point-level loss causing it to never converge. Further, the loss function is not directly differentiable with respect to the output of LCFCN. A set of pseudo ground-truth labels are generated every time the soft probabilities are output from the network. Those hard labels are then used to update the model parameters. In this case, it is important to perform early stopping on a validation set before evaluating the model on the test set.

The optimization speed and convergence is also affected by the amount of weighing done between the 4 loss terms. If there are many overlapping objects, then it might help to lower the weight for the split-level loss. Otherwise, the amount of background pixels might be much more than foreground pixels making LCFCN predict most regions as background leading to many false negatives.

It is easier for LCFCN to locate larger objects than smaller ones. LCFCN has more space to output a blob around the center of larger objects, even if these objects overlap. In this case, the split-level loss is easier to optimize. For smaller objects, it might be difficult to split the blobs because typical backbones have limited capacity. The receptive field of the backbone and the amount of downsamplings required might not allow LCFCN to distinguish between small objects that are few pixels apart.

2.6 Conclusion

We propose LC-FCN, a fully-convolutional neural network, to address the problem of object counting using point-level annotations only. We propose a novel loss function that encourages the model to output a single blob for each object instance. Experimental results show that LC-FCN outperforms current state-of-the-art models on the PASCAL VOC 2007, Trancos, and Penguins datasets which contain objects that are heavily occluded.

Chapter 3

Instance Segmentation with Point Supervision

Instance segmentation methods often require costly per-pixel labels. We propose a method that only requires point-level annotations. During training, the model only has access to a single pixel label per object, yet the task is to output full segmentation masks. To address this challenge, we construct a network with two branches: (1) a localization network (L-Net) that predicts the location of each object; and (2) an embedding network (E-Net) that learns an embedding space where pixels of the same object are close. The segmentation masks for the located objects are obtained by grouping pixels with similar embeddings. At training time, while L-Net only requires point-level annotations, E-Net uses pseudo-labels generated by a class-agnostic object proposal method. However, since it is challenging to learn object masks from point-level annotations only, the object proposal method was trained on a large scale dataset labeled with class-agnostic masks. We evaluate our approach on PASCAL VOC, COCO, KITTI and CityScapes datasets. The experiments show that our method (1) obtains competitive results compared to fully-supervised methods in certain scenarios; (2) outperforms fully- and weakly supervised methods with a fixed annotation budget; and (3) is a first strong baseline for instance segmentation with point-level supervision.

3.1 Introduction

Instance segmentation is the task of classifying every object pixel into a category and discriminating between individual object instances. It has a wide variety of applications such as autonomous driving [26], scene understanding [75, 35], and medical imaging [94].

Most instance segmentation methods, such as Mask-RCNN [49] and MaskLab [18], rely on per-pixel labels which requires huge human effort. For instance, obtaining labels for PASCAL VOC [35] requires an average time of 239.7 seconds per image [10]. Other datasets with more objects to annotate such as CityScapes [26] can take up to 1.5 hours per image.

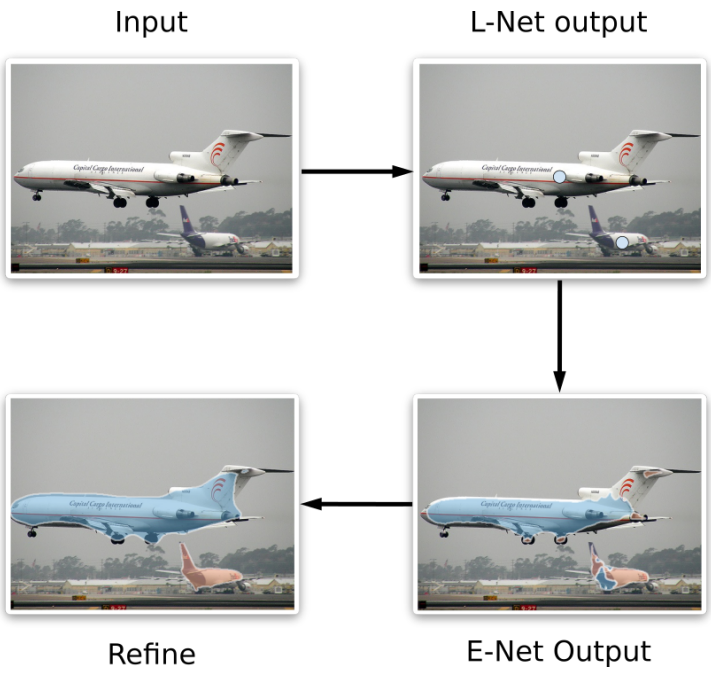


Figure 3.1: **WISE network.** Our method, WISE, is trained using point-level annotations only. At test time, WISE first uses L-Net to locate the objects in the image, and then uses E-Net to predict the masks of the located objects. Finally, the predicted masks are refined with the help of an object proposal method.

Indeed, having a method that can train with weaker supervision can vastly reduce the required annotation cost. According to Bearman et al. [10], manually collecting image-level and point-level labels for the PASCAL VOC dataset took only 20.0 and 22.1 seconds per image, respectively. These annotation methods are an order of magnitude faster than acquiring full segmentation labels (see Figure 3.2 for a comparison between the point-level and per-pixel annotation methods).

For semantic segmentation, other forms of weaker labels were explored such as bounding boxes [57], scribbles [74], and image-level annotation [141]. For instance segmentation, few work exist that use weak supervision [141, 24]. In this paper, we propose a weakly supervised Instance SEgmentation (WISE) network, which is the first to address this task using point-level supervision and supervision from pretrained proposal networks. However, since it is challenging to learn object masks from point-level annotations only, the object proposal method was trained on a large scale dataset labeled with class-agnostic masks.

WISE has two branches: (1) a localization network (L-Net) that predicts the

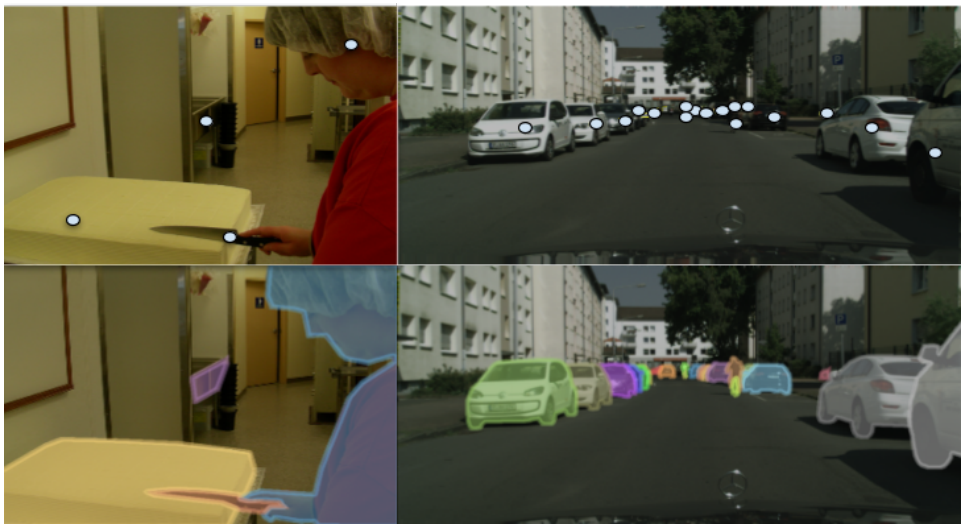


Figure 3.2: **Image annotation.** Point-level (top) and per-pixel (bottom) labels for COCO and the CityScapes datasets.

location of each object and (2) an embedding network (E-Net) that learns an embedding space where pixels of the same object are closer. L-Net is trained using a loss function that forces the network to output a single point per object instance. E-Net is trained using a similarity-based objective function to force the pixel embeddings to be similar within the same object mask. Since we do not have access to the ground-truth object masks, we instead use pseudo-labels generated by an object proposal method. These pseudo-labels belong to arbitrary objects and have no class labels and therefore cannot be directly applied for instance segmentation. At test time, L-Net first predicts the object locations. Second, E-Net outputs the embedding value for each pixel. Then the pixels with the most similar embeddings to an object's predicted pixel location become part of that object's mask (Figure 3.1).

We summarize our contributions as follows: (1) we provide a first strong baseline for instance segmentation with point-level supervision; (2) we evaluate our method on a wide variety of datasets, including, PASCAL VOC [35], COCO [75], CityScapes [26], and KITTI [41] datasets; (3) we obtain competitive results compared to fully-supervised methods; and (4) our method outperforms fully- and weakly supervised methods when the annotation budget is limited.

3.2 Related Work

Our approach lies at the intersection of object localization, metric learning, object proposal methods, and instance segmentation. These topics have been studied extensively and we review the literature below. The novelty of our method is the combination of these techniques into a new setup, namely, instance segmentation with point-level supervision.

Instance segmentation. Instance segmentation is an important computer vision task that can be applied in many real-life applications [102, 104]. This task consists of classifying every object pixel into categories and distinguishing between object instances. Most methods follow a two step procedure [49, 18, 38], where they first detect objects and then segment them. For instance, Mask-RCNN [49] uses Faster-RCNN [103] for detection and an FCN network [79] for segmentation. However, these methods require dense labels which leads to a high annotation time for new applications.

Embedding-based instance segmentation. Another class of instance segmentation methods obtain the object masks by grouping pixels based on a similarity measure. Notable works in this category include methods based on watershed [9], template matching [123] and associative embedding [86]. Fathi et al. [36] propose a grouping-based method that first learns the object locations and then learns the pixel embeddings in order to distinguish between object instances. These methods also require per-pixel labels which are costly to acquire for new applications. However, our method follows a similar procedure for obtaining the segmentation masks while requiring weaker supervision.

Weakly supervised instance segmentation. Per-pixel labels used by fully supervised instance segmentation methods require a high annotation cost [35, 26]. Therefore many weakly supervised methods have been explored for object detection [120, 12], semantic segmentation [90, 60, 1, 109] and instance segmentation [57, 141, 24]. Point-level annotation is one of the fastest ways to annotate object instances. Although such annotation one of the least informative forms of weak supervision, they were shown to be effective for semantic segmentation [10]. Inspired by their cost-effectiveness, we explore the novel problem setup of instance segmentation with point supervision in this work.

Object localization with point supervision. An important step in instance segmentation is to locate objects of interest before segmenting them. One way to perform object localization is to use object detection methods [103, 99]. However, these methods require bounding-box labels. In contrast, several methods exist that use weaker supervision to identify object locations [116, 117, 68, 71]. Close to our

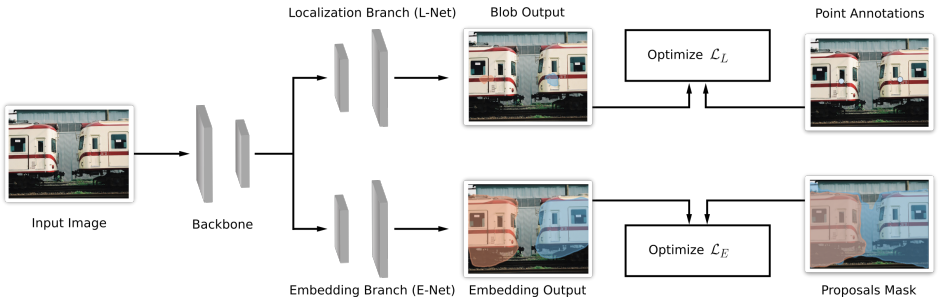


Figure 3.3: **Training WISE.** Our method consists of a localization branch (L-Net) and an embedding branch (E-Net). During training, L-Net optimizes Eq. **??** in order to output a single point per object instance. E-Net optimizes Eq. 3.2 in order to group pixels that belong to the same object instance.

work is LCFCN (Chapter 2) which uses point-level annotations in order to obtain the locations and counts of the objects of interest. While this method gives accurate counts and identifies a partial mask for each instance, it does not produce accurate segmentation of the instances. We extend this method by using an embedding network that groups pixels that are most similar to the predicted object locations in order to obtain their masks.

Object proposals. Weakly supervised methods often rely on object proposals [51] to ease the task of detection [120, 12] and segmentation [90, 10, 141, 60]. Object proposal methods can output thousands of object candidates per image, but they do not output class labels for these candidates. Many such methods have been proposed over the last decade in order to obtain high quality candidates [124, 143, 4, 82, 91, 92]. SharpMask [92] is a popular deep-learning based object proposal method that has been successfully applied to many weakly supervised computer vision problems. However, their output object masks cannot be directly used for instance segmentation as they belong to arbitrary objects and have no class labels. Our framework uses pseudo-labels generated by SharpMask.

3.3 Proposed Method

We address the problem of weakly supervised instance segmentation, where each labeled object has a single point annotation. Our method, WISE network, has two output branches that share a common feature extraction backbone (Figure 3.3): (1) a localization branch (L-Net) that is trained for locating objects in the image, and (2) an embedding branch (E-Net) that outputs an embedding vector for each pixel.

L-Net is trained using point-level annotations in order to output a single pixel for each object to represent its location and category in the image. On the other hand, E-Net is trained using masks obtained by a pretrained proposal method. The trained E-Net can output an embedding vector for each pixel such that similar ones belong to the same object's mask. Note that proposal methods have been widely used for different weakly supervised problem setups [141, 24, 12, 90, 10]

WISE obtains the mask of an object as follows. First, L-Net outputs a pixel label per object to identify its location, category, and instance. Then, the embedding of every pixel in the image is compared to the embedding of the pixels predicted by L-Net to identify which object instance they belong to. Finally, the pixels are grouped to form the object masks in the image.

3.3.1 Localization Branch (L-Net)

The goal of L-Net is to obtain the locations and categories of the objects in the image. L-Net is based on LC-FCN Chapter 2 which trains with point level annotations to produce a single blob per object. While LC-FCN was originally designed for counting, it is able to locate objects effectively. LC-FCN is based on a semantic segmentation architecture that is similar to FCN [79]. Indeed, semantic segmentation methods are not suitable for instance segmentation as they often predict large blobs that merge several object instances together. LC-FCN addresses this issue by optimizing a loss function that ensures that only a single small blob is predicted around the center of each object. The location loss term $\mathcal{L}_L$ is described in detail in Chapter 2.

Since LC-FCN's predicted blobs are too small to be considered as useful segmentation masks, we instead leverage the location of each blob by identifying the pixel with the highest probability of being foreground (Figure 3.4).

3.3.2 Embedding Branch (E-Net)

The goal of E-Net is to produce object masks by grouping pixels with similar embeddings together. E-Net's architecture is based on FCN8 [79], which can output an embedding vector per image pixel. Using a similarity loss, E-Net learns to output similar embeddings for pixels that belong to the same object and dissimilar otherwise. This loss requires several points per object (including the background) in order to distinguish between different objects. While we do not have access to the ground-truth masks, we instead use masks generated by an object proposal method to assign a mask for each object.

E-Net learns a mapping from an input image to a set of embedding vectors of size d for each pixel. Let E_i and E_j be the embeddings for pixel i and pixel j,

Figure 3.4: **Localization branch (L-Net).** L-Net's raw output is a small blob per predicted object (top). L-Net's final output is the set of pixels with the largest activation within their respective blobs (bottom). These pixels are used as input to E-Net at test time.

respectively. We measure the similarity between a pair of pixels using a squared exponential kernel function, similar to that of Fathi et al. [36]:

$$S(i,j) = \exp\left(-\frac{||E_i - E_j||_2^2}{2d}\right), \tag{3.1}$$

where $S(E_i, E_j)$ tends to 1 as E_i and E_j get closer, and tends to 0 as they get apart in the embedding space. Note that our method can use other similarity functions as in Newell et al. [86], Fathi et al. [36], Kong and Fowlkes [61].

Our goal is to train E-Net such that embeddings of pixel pairs belonging to the same object instance $y_i = y_j$ have the same embedding $S(i,j) = 1$, and to different object instances $y_i \neq y_j$ have different embeddings $S(i,j) = 0$. Therefore, E-Net minimizes the following loss function[4]:

$$\mathcal{L}_E = -\sum_{(i,j)\in P}\left[\mathbb{1}_{\{y_i=y_j\}}\log S(E_i,E_j) + \mathbb{1}_{\{y_i\neq y_j\}}\log\left(1 - S(E_i,E_j)\right)\right], \tag{3.2}$$

where P is a set of pixel pairs. We found that using the negative log-liklihood gave us a more stable optimization compared to using the L2 loss.

[4]Note that the log and exp cancel out in the first term of the equation but not the second term.

Figure 3.5: **pseudo-labels**. (Left) ground-truth point-level annotations; (Center) a set of generated object proposals that intersect with the point annotations; (Right) proposals with best "objectness".

Since we require more than one point label per object to optimize Equation 3.2, we use extra points from pseudo-labels generated by an object proposal method (see Figure 3.5). We used the SharpMask proposal method which is a fully-convolutional neural network that was trained to output class agnostic masks [92]. At each training iteration, the mask pseudo-label of an object is randomly selected from the set of proposals (obtained by the proposal method) that intersect with the object's point annotation. Further, we define the background as the region that does not contain any proposal mask.

We obtain the set of pixel pairs P for Eq. 3.2 as follows. We pair each pixel represented by the point-level annotation with k random pixels[5] from each object's mask pseudo-label including the background region. This randomness allows the model to learn the important pixels that correspond to the objects of interest. The final objective function of WISE is defined as:

$$\mathcal{L}_W = \lambda \cdot \mathcal{L}_L + (1 - \lambda) \cdot \mathcal{L}_E \ , \tag{3.3}$$

where λ is the weight that balances between L-Net's and E-Net's loss terms.

3.3.3 Prediction at Test Time

As shown in Figure 3.6, WISE predicts masks of objects using the following steps. First, L-Net outputs a pixel coordinate for each object representing its location and category. Second, E-Net outputs the embedding vectors for all pixels in the image. Third, E-Net computes the similarity (Equation 3.1) between each pixel in the image and two sets of pixels: (1) L-Net's predicted pixel coordinates, and (2) several selected background pixels. Next, E-Net assigns each pixel to the most similar object, resulting in a mask for each object including the background region.

[5] We chose k as the number of objects in the image.

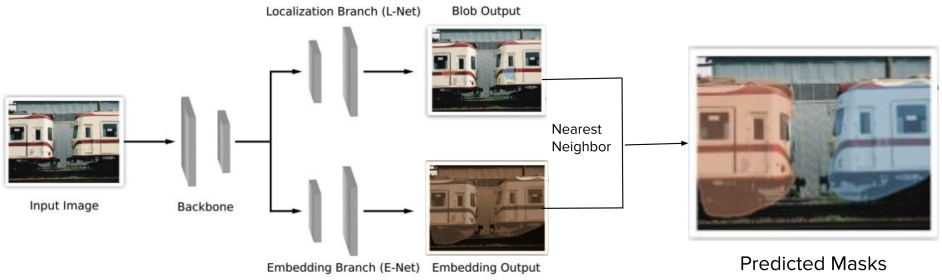


Figure 3.6: **Prediction**. First L-Net outputs blobs for the objects of interest. Second, E-Net outputs embeddings for each pixel in the image. Then, nearest-neighbors is applied to group pixels based on their similarity in the embedding space. The grouped pixels form the masks of the objects of interest.

Finally, the object masks are refined by replacing them with the mask pseudo-label (generated from a proposal method) with the largest Jaccard similarity (see Figure 3.1).

For selecting the background pixels deterministically, we first define the background regions as the pixels that do not correspond to any of the generated proposal masks. We use the k-means algorithm for clustering the pixel embeddings into k groups. Then, for each cluster we select the closest pixel to the mean of that cluster, giving us k representative pixels from the background.

3.4 Experiments

We evaluate the WISE network on a wide variety of datasets: PASCAL VOC [35], COCO [75], CityScapes [26], and KITTI [41] datasets. We compare our results against fully-supervised and weakly supervised methods. We compare WISE against several baselines to showcase the efficacy of each of its components. We also fix the annotation budget for acquiring per-pixel, point-level, and image-level labels and compare several models based on the type of label they require. Unless otherwise specified, the performance is measured using average precision (AP) as in He et al. [48], computed with Intersection-over-Union (IoU) thresholds of 0.25, 0.5, and 0.75.

3.4.1 Methods and Baselines

We include the following methods in our benchmarks:

Method	$\mathbf{AP}_{25}$	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$
L-Net + Blobs	08.4	01.2	00.1
L-Net + Best proposal	42.9	33.4	19.1
L-Net + Oracle proposal	57.3	45.1	37.2
L-Net + GT-Mask	61.2	61.2	61.2
PRM + E-Net	43.0	32.0	19.0
GT-points + E-Net	63.1	47.0	26.3
WISE (L-Net + E-Net)	53.5	43.0	25.9

Table 3.1: **Ablation Studies**. A benchmark illustrating the contribution of each WISE's component on PASCAL VOC 2012.

L-Net + Blobs: use the raw output of L-Net (see Figure 3.4) (which is a predicted blob per object in the scene) as mask prediction.
L-Net + Best proposal: replace each object location predicted by L-Net with the SharpMask's proposal that has the highest "objectness" score.
L-Net + Oracle proposal: replace each object location predicted by L-Net with the SharpMask's proposal that achieves the highest evaluation score (e.g.mAP).
L-Net + GT-Mask: replace each object location predicted by L-Net with the ground-truth mask.
PRM + E-Net: use the object locations predicted by PRM (as described in Zhou et al. [141]) as input to E-Net to obtain the object masks. Note that PRM only requires image-level labels.
GT-points + E-Net: use the ground-truth object locations (point-level annotations) as input to E-Net to obtain the object masks.
WISE (L-Net + E-Net): use L-Net's predicted object locations as input to E-Net to obtain the object masks.

3.4.2 Implementation Details

L-Net and E-Net share the same backbone, a ResNet-50 [48] pretrained on ImageNet [31]. They also have independent upsampling paths with similar architecture as FCN8 [79]. The number of output channels for L-Net is the number of classes, and for E-Net is $d = 64$, the size of a pixel's embedding vector. We observed minor differences in the results between different embedding dimensions. For each image, we use 1000 pretrained SharpMask [92] proposals (note that we do not fine-tune the proposal network on any dataset, but it has been pretrained on classes that are present in our training set). During training, for each point-annotation we sam-

Method	Annotation	AP_{25}	AP_{50}	AP_{75}
Mask R-CNN [142]	per-pixel	17.1	11.2	03.4
SPN [142]	image-level	26.0	13.0	04.0
PRM [141]	image-level + proposals	44.0	27.0	09.0
Cholakkal et al. [24]	image-level + proposals	**48.5**	30.2	14.4
PRM + E-Net (Ours)	image-level + proposals	43.0	32.0	19.0
WISE (Ours)	point-level + proposals	47.5	**38.1**	**23.5**

Table 3.2: **PASCAL VOC 2012 with a fixed annotation budget.** Comparison across methods with the same annotation budget.

Method	plane	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
SDS [47]	58.8	0.5	60.1	34.4	29.5	60.6	40.0	73.6	6.5	52.4	31.7	62.0	49.1	45.6	47.9	22.6	43.5	26.9	66.2	66.1	43.8
Chen et al. [21]	63.6	0.3	61.5	43.9	33.8	67.3	46.9	74.4	8.6	52.3	31.3	63.5	48.8	47.9	48.3	26.3	40.1	33.5	66.7	67.8	46.3
PFN [72]	76.4	15.6	74.2	54.1	26.3	73.8	31.4	92.1	17.4	73.7	48.1	82.2	81.7	72.0	48.4	23.7	57.7	64.4	88.9	72.3	58.7
R2-IOS [73]	87.0	6.1	90.3	67.9	48.4	86.2	68.3	90.3	24.5	84.2	29.6	91.0	71.2	79.9	60.4	42.4	67.4	61.7	94.3	82.1	66.7
Fathi et al. [36]	69.7	1.2	78.2	53.8	42.2	80.1	57.4	88.8	16.0	73.2	57.9	88.4	78.9	80.0	68.0	28.0	61.5	61.3	87.5	70.4	62.1
WISE (Ours)	59.0	5.6	63.6	41.4	21.9	40.6	34.1	73.8	8.5	38.7	29.1	64.6	58.1	60.4	33.3	25.1	43.8	32.7	64.7	60.7	43.0

Table 3.3: **Comparison to fully supervised methods.** Per-class comparison against the AP_{50} metric on PASCAL VOC 2012.

ple a proposal non-uniformly based on its "objectness" score to represent its mask pseudo-label. We set k as the number of predicted objects (by L-Net) for selecting the background pixels at test time. The model is trained using the Adam [58] optimizer with a learning rate of 10^{-5} and a weight decay of 0.0005 for $200k$ iterations with a batch size of 1. We choose $\lambda = 0.1$ in Equation 3.3 in order to make the scale between its two loss terms similar.

3.4.3 Experiments on PASCAL VOC 2012

PASCAL VOC 2012 [35] contains $1,464$ and $1,449$ images for training and validation respectively, where objects come from 20 categories. We use the point-level annotations provided by Bearman et al. [10] as ground-truth for training our methods. We report the AP across several thresholds on the validation set, as described in the dataset's instance segmentation setup [35].

Comparison to methods and baselines.

In this section, we discuss the results shown in Table 3.1. A straightforward method to obtain object masks is to use L-Net's raw output (which we refer to as "L-Net +

Model	COCO 2014			KITTI			CityScapes		
	$\mathbf{AP}_{25}$	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$	$\mathbf{AP}_{25}$	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$	$\mathbf{AP}_{25}$	$\mathbf{AP}_{50}$	$\mathbf{AP}_{75}$
L-Net Best proposal	18.3	13.6	7.3	46.4	38.1	22.2	27.2	15.5	6.7
WISE (Ours)	25.8	17.6	7.8	63.4	49.8	30.9	28.7	18.2	8.8

Table 3.4: **Baseline comparisons.** Results across different average precision IoU thresholds.

Blobs"). However, it performs poorly as the predicted blobs are often small around the center of the object.

A natural extension is to replace L-Net's predicted blobs by a segment proposal obtained from an object proposal method. Therefore, we discovered a reasonable strategy which is to replace each of L-Net's predicted blobs by the proposal of highest "objectness" score ("L-Net + Best-proposal"). However, "L-Net + Oracle" shows that a perfect proposal selection strategy can vastly improve on the segmentation results.

Accordingly, we propose WISE which improves on "L-Net + Best-proposal" by having E-Net that learns rough segmentation of the objects. This method selects better proposals by choosing those with the highest intersection over union (IoU). Note that other object proposal selection strategies have been used in other weakly supervised instance segmentation setups [141, 24].

To assess how much improvement we can make over L-Net, we report the results of "GT-points + E-Net" which uses the ground-truth points instead of L-Net's predictions. We see that L-Net's performance is close to its upper-bound. Further, we provide the results of "PRM + E-Net" which is an extension to WISE that can train using image-level annotations only. Similarly, we observe that the results are not widely different. However, image-level labels might not be suitable for datasets when the number of objects in an image is high and when the same object class exist in almost every image as the *car* category in CityScapes.

Comparison to Similar Annotation Time

We compare the performance between state-of-the-art methods in Table 3.2 when the annotation time is fixed. Therefore, we limit the annotation budget to around 8.13 hours which is calculated as $20.0 \times 1,464$ seconds. Bearman et al. [10] has shown that it takes 20.0, 22.1, and 239.7 seconds per image for collecting image-level, point-level, and per-pixel labels, respectively. As a result, for the same annotation time budget, we acquire $1,464$ images with image-level labels, $1,325$ images with point-level labels, and 122 images with per-pixel labels. We selected these images uniformly without replacement from the training set. We also reported the result of Mask R-CNN [84] trained on the images with the per-pixel labels. The ta-

ble shows that our method significantly outperforms other approaches, suggesting that using point-level annotations is a cost-effective labeling method for instance segmentation. Further, Figure 3.7 illustrates that WISE can capture high quality masks for PASCAL VOC objects, although it can fail in merging two masks of the same object such as in the horse image.

Comparison to Weakly and Fully Supervised Methods

Acquiring point-level labels is almost as cheap as image-level labels, yet they vastly improve results, as shown in Table 3.2. For a fair evaluation, we compare "PRM + E-Net" which uses image-level labels against current state-of-the-art image-level instance segmentation methods. The concurrent work of Cholakkal et al. [24] performs better with respect to AP_{25}, which is expected as their counting results is better than LCFCN which is what L-Net is based on.

Further, we report WISE results against fully supervised methods in Table 3.3 for each category with respect to AP_{50}. While WISE achieves competitive results, there is room for improvement between weakly and strong- supervised methods.

Model	**AP**$_{50}$	**AP**$_{75}$
Base-DA [34]	46.0	28.1
Mask-RCNN [49]	55.2	35.3
WISE (Ours)	17.4	07.7

Table 3.5: **COCO 2014.** Comparison to fully supervised methods.

3.4.4 Experiments on COCO 2014

For COCO 2014 [75], we train on the union of the 80k train images and the 35k subset of validation images, and report the results on *minival* consisting of 5k images, following the experimental setup of He et al. [49]. It consists of 80 categories belonging to a wide variety of everyday objects. We obtain ground-truth points by taking the pixel with the largest distance transform for each instance segmentation mask. We use the standard COCO metrics including AP (averaged over IoU thresholds), AP_{50}, and AP_{75}. Table 3.4 shows that WISE outperforms our baseline "L-Net + Best Proposal", which suggests that E-Net generates better proposal masks. The qualitative results in Figure 3.7 show that WISE can successfully capture the mask of diverse objects. Table 3.5 shows that while our results are poor compared to fully supervised methods, they establish a first strong baseline for instance segmentation with point-level supervision.

3.4.5 Experiments on KITTI

KITTI [41] is a meaningful benchmark for autonomous driving. Using the setup described in Zhang et al. [138], we train our models on the $3,712$ training images where the ground-truth points are the provided bounding box centers. We reported results on the 120 validation images using the *MUCov* and *MWCov* metrics, as described in Silberman et al. [112]. Table 3.4 shows that WISE significantly outperforms the baseline "L-Net + Best Proposal", suggesting that relying on the best "objectness" score for picking the proposal is not the optimal approach. Furthermore, Table 3.6 shows that WISE achieves competitive results compared to methods that use full supervision. Figure 3.7 shows quality masks being generated for the cars and persons objects on KITTI images by WISE.

Model	**MWCov**	**MUCov**
DepthOrder [137]	70.9	52.2
DenseCRF [138]	74.1	55.2
AngleFCN+Depth [123]	79.7	75.8
Recurrent+attention [102]	80.0	66.9
WISE (Ours)	74.2	58.9

Table 3.6: **KITTI.** Comparison to fully supervised methods.

3.4.6 Experiments on CityScapes

CityScapes [26] is a popular autonomous driving benchmark for instance segmentation. It contains $2,975$ high-resolution training images, and 500 validation images that represent street scenes acquired from an on-board camera. The pixels are labeled into 19 classes, but only 8 classes belong to countable objects (used for instance segmentation): person, rider, car, truck, bus, train, motorcycle, and bicycle. The ground-truth point for each object is the pixel with the largest distance transform within its corresponding ground-truth segmentation mask.

Table 3.4 shows that WISE sets a new strong baseline for the weakly supervised setting, while achieving better results than the comparable baseline "L-Net + Best proposal". Further, Figure 3.7 illustrates that our method can obtain good masks for various objects of interest. However, fully supervised methods shown in Table 3.7 outperform our weakly supervised method with a large margin, motivating future research on this problem setup.

In Table 3.8, we compare "GT-points + E-Net" against the methods proposed by Remez et al. [101] which use bounding box ground-truth labels at test time.

Using their evaluation setup, we report the results in Table 3.8 which shows better results across four categories. This is despite E-Net using weaker labels than Cut & Paste. According to Bearman et al. [10], it takes an average of 10.2 seconds to acquire a bounding box, but only 2.4 seconds to get an annotation for a single object instance.

Method	**AP**
InstanceCut [59]	15.8
DWT [9]	19.8
SGN [77]	29.2
Mask-RCNN [49]	31.5
WISE (Ours)	07.8

Table 3.7: **CityScapes.** Comparison to fully supervised methods.

Method	**Car**	**Person**	**T. light**	**T. sign**
Box [101]	62.0	49.0	76.0	76.9
Simple Does it [57]	68.0	53.0	60.0	51.0
GrabCut [106]	62.0	50.0	64.0	65.0
Cut & Paste [101]	67.0	54.0	77.0	79.0
Fully Supervised [101]	80.0	61.0	79.0	81.0
GT-points + E-Net (Ours)	77.6	55.4	77.8	80.1

Table 3.8: **CityScapes.** Methods with bounding boxes at test time.

3.5 Limitations

The two-stage procedure of WISE-Net might cause error propagation. Errors from L-Net can propagate to E-Net which can cause a drop in performance. Thus, we plan to improve on this architecture by investigating single-stage methods that can learn to directly perform segmentation for the objects of interest.

Another limitation is that the performance of the embedding network is heavily influenced by the quality of the masks output by the proposal network. It is challenging to learn object masks from point-level annotations only, but we plan to investigate methods based on self-attention to address this challenging setup.

3.6 Conclusion

In this chapter, we have introduced a weakly supervised instance segmentation network (WISE). It can train by using point-level annotations and by leveraging pseudo-labels from object proposal methods. WISE uses L-Net to first detect the object locations which are then given as input to E-Net in order to obtain the segmentation masks. E-Net is based on an embedding network that groups pixels in the image-based on their similarity which are then used to select the best matching proposal mask. We have validated our method across a wide variety of datasets. The results show that WISE obtains competitive results against fully supervised methods and outperform weakly supervised methods with a fixed annotation cost. The results also provide a strong first baseline for instance segmentation with point-level supervision. Although a pretrained proposal method was used in this problem setup, it was not finetuned on any of our datasets. However, an interesting future direction is to address this task with a more challenging setup that requires proposal-free methods.

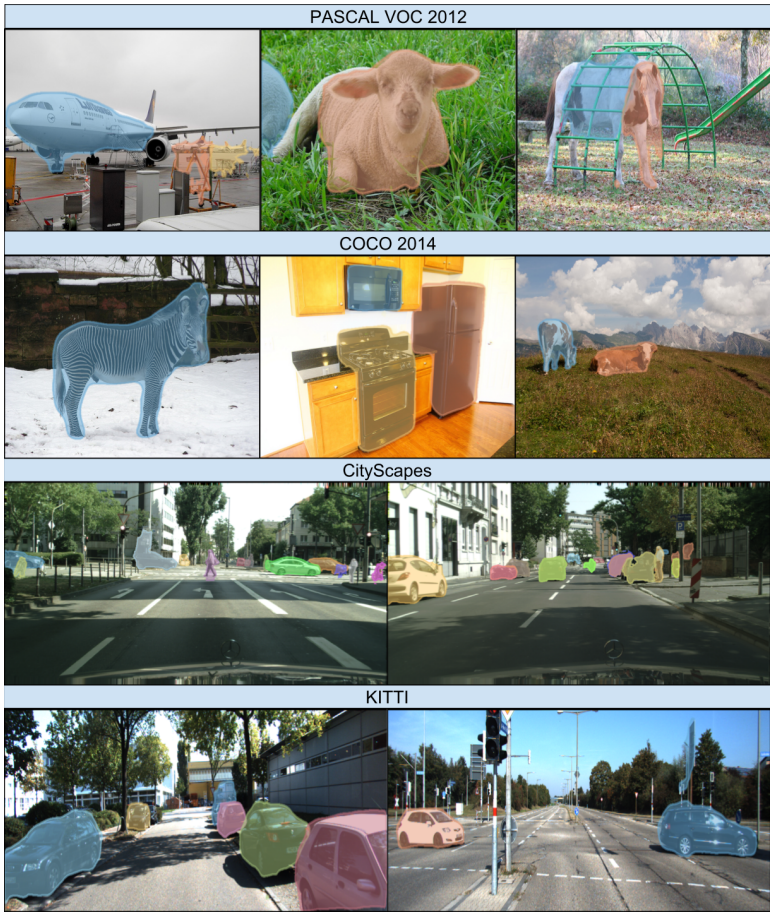


Figure 3.7: **Qualitative results.** Qualitative results of WISE on the four datasets evaluated.

Chapter 4

Where are the Masks

A major obstacle in instance segmentation is that existing methods often need many per-pixel labels in order to be effective. These labels require substantial human effort and for certain applications, such labels are not readily available. To address this limitation, we propose a novel framework that can effectively train with image-level labels, which are significantly cheaper to acquire. For instance, one can do an internet search for the term ”car” and obtain many images where a car is present with minimal effort. Our framework consists of two stages: (1) train a classifier to generate pseudo masks for the objects of interest; (2) train a fully supervised Mask R-CNN on these pseudo masks. Our two main contribution are proposing a pipeline that is simple to implement and is amenable to different segmentation methods; and achieves new state-of-the-art results for this problem setup. Our results are based on evaluating our method on PASCAL VOC 2012, a standard dataset for weakly supervised methods, where we demonstrate major performance gains with respect to mean average precision compared to existing methods.

4.1 Introduction

The recent progress in deep neural networks (DNNs) and segmentation frameworks has given us major improvements in the task of instance segmentation [49, 18]. Their success has been demonstrated in various applications such as autonomous driving [26], scene understanding [75, 35], and medical imaging [94, 62]. Nonetheless, these methods require a large number of training examples with per-pixel labels, or labels which distinguish between object categories and instances in the image. As acquiring them is often prohibitively expensive, the effectiveness of these methods is limited to a small range of datasets and object categories.

Many weakly supervised methods emerged to overcome the need for per-pixel labels. Instead, they only require weaker labels ranging from bounding boxes [57], scribbles [74], and image-level [141, 24, 2] annotations. This makes acquiring datasets a significantly more scalable endeavour. According to Bearman et al. [10], it requires 20 sec/img to acquire image-level labels (which are labels that only indicate whether an object class appears in an image) for PASCAL VOC [35],

compared to 239.7 sec/img for acquiring per-pixel labels. To date, only two weakly supervised methods address instance segmentation with image-level labels, making our work one of the few that tackles a relatively unexplored research area.

Perhaps the first work to address this problem setup is PRM [141]. It trains a classifier which can then identify local regions belonging to different objects of the same category. It extends CAM-based methods [109, 1] by not only identifying large regions where objects are vaguely located, but also identifying peaks that represent the specific locations of the object instances. At test time, the trained PRM obtains the object masks in two steps. First, it uses the gradient with respect to the input to get a rough mask of the objects using a process called peak backpropagation. This results in a peak response map. Then, the masks in this map are replaced by the best matching proposal masks, which are generated from a pretrained object proposal method [4, 93]. However, PRM's results are much worse than that of fully supervised methods, leaving a large room for improvement (Table 4.1).

We propose WISE-ILS, a Weakly supervised Instance SEgmentation method that only requires Image-Level Supervision. It builds on PRM by using its output pseudo masks to train a fully-supervised method, namely, Mask R-CNN [84]. Our intuition as to why this procedure is effective is that Mask R-CNN is potentially robust to noisy pseudo masks, and the noisy labels within these masks might be ignored during training as they are potentially uncorrelated. The success of such a de-noising strategy has been demonstrated in semantic segmentation and object localization [57].

We show that simple techniques for obtaining the pseudo masks lead to a surprisingly effective supervision for Mask R-CNN. We summarize our contributions as follows. (1) We present a novel framework that can effectively train a fully supervised method on pseudo mask labels obtained from image-level class labels; (2) we show that our framework is amenable to different localization and segmentation methods (for example, a density-based PRM [24] can be used for localization and RetinaMask [38] can be used for instance segmentation), and (3) we achieve new state-of-the-art results on a standard weakly supervised instance segmentation benchmark.

4.2 Related Work

Instance segmentation is widely studied within the computer vision community [49, 18, 38]. However, an ongoing challenge is that it is time-consuming and expensive to obtain the required per-pixel labels needed by most instance segmentation methods [35, 26]. Current trends to overcome this issue leverage weaker labels (such as image-level labels) and pseudo labels obtained with the help of object proposal

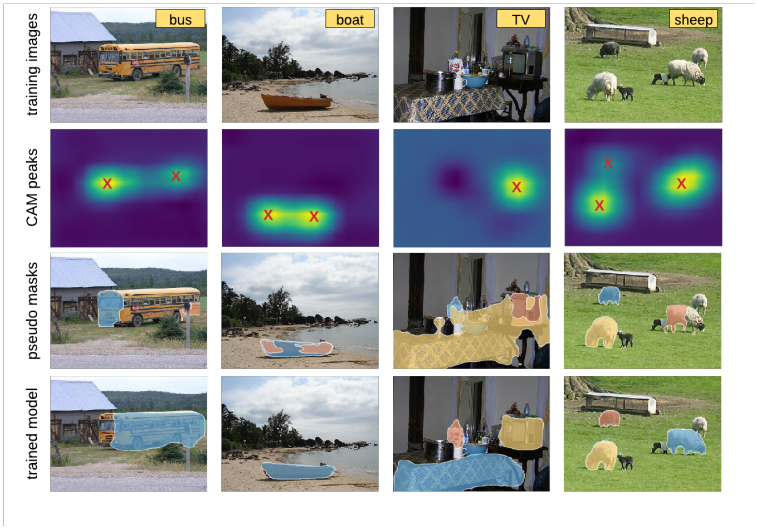


Figure 4.1: **Framework overview.** Our weakly supervised instance segmentation (WISE-ILS) method learns to perform instance segmentation with image-level supervision. First, a classifier is trained with a peak stimulation layer to identify peaks at which the objects are located (row 2). A proposal gallery (such as MCG [4]) is used to obtain rough masks for the located objects, which are then used as pseudo masks to train Mask R-CNN [49] (row 3). Row 4 shows the output of a Mask R-CNN trained on the noisy pseudo mask labels.

methods. While most of these methods are for object detection and semantic segmentation, we review them below as they are relevant.

Instance segmentation. Instance segmentation is one the most challenging tasks in computer vision. The task is to classify every object pixel into its corresponding category and distinguish between object instances [102, 104]. Most recent methods rely on deep networks and follow a two step procedure [49, 18, 38], where they first detect objects and then segment them. For instance, Mask-RCNN [49] uses Faster-RCNN [103] for detection and an FCN network [79] for segmentation. In this work, we use Mask R-CNN as our fully supervised method and train it on pseudo masks instead of the costly per-pixel labels.

Learning with weak supervision. Due to the taxing task of acquiring per-pixel labels, many weakly supervised methods emerged that can leverage labels that are much cheaper to acquire [35, 26]. These labels range from bounding boxes [57], scribbles [74], points [10, 65, 66], and image-level annotation [141]. Our setup considers one of the weakest forms of annotation, image-level labels.

Image-level labels as weak supervision. Acquiring image-level labels is an attractive form of annotation due to its extremely cheap cost. The annotator only needs to indicate whether a certain object class appears in an image, regardless of how many of them appear. While this form of annotation has gained steam within the research community, most of the proposed methods are for semantic segmentation [48, 120, 1]. Perhaps the lack of such research for instance segmentation is partially accounted for by the fact that instance segmentation is a more challenging task. Only recently did works emerge for this problem setup [141, 24, 2]. Zhou et al. [141] and Cholakkal et al. [24] extend the Class Activation Map (CAM) [140], by not only identifying a heatmap that vaguely represents the regions where objects are located, but also identifying peaks of that heatmap that represent the locations of different objects. At test time, they adopt a post-processing step that matches each located object with a proposal, generated from an object proposal method. These proposals are considered as the final instance segmentation output. In contrast, we use these outputs as pseudo masks to train a fully supervised method.

Learning with pseudo labels. Our method first generates pseudo-labels and then train a model on these labels in a fully-supervised manner. While this is novel for instance segmentation, similar approaches were used for object detection [119] and semantic segmentation [27, 96, 57] in weakly supervised settings. However, these methods cannot be directly applied to instance segmentation, as they do not distinguish between object instances. Many such methods also rely on object proposals [51] to ease the task of detection [120, 12], and segmentation [90, 10, 141, 60]. Object proposals are class-agnostic methods that can output thousands of object candidates per image and have progressed significantly over the last decade [124, 143, 4, 82, 91, 93]. Similar to PRM [141] and PRM+Density [24], we also leverage object proposals to generate the pseudo masks.

4.3 Proposed Method

Our approach to instance segmentation with image-level supervision consists of two main steps: (1) obtain pseudo masks for the training images given their ground-truth image-level labels; and (2) train a fully supervised instance segmentation method on these pseudo masks (Figure 4.2). In particular, this framework is based on two components: a network that obtains the pseudo masks by training a PRM [141] on the image-level labels and leveraging object proposal methods [4] and Mask R-CNN [49] as a fully supervised instance segmentation method. We show the training steps of our framework in Algorithm 1.

At test time, we can predict the object instance masks using the trained Mask R-CNN only, discarding the PRM component. In this setup, we are interested in

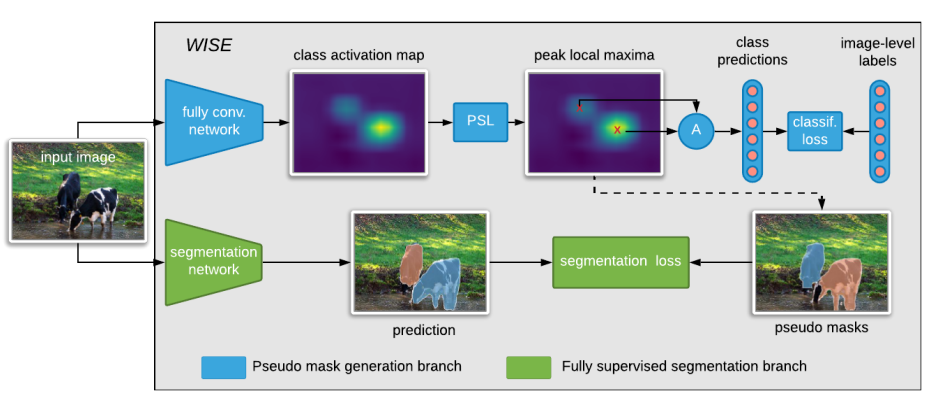


Figure 4.2: **WISE-ILS training.** The first component (shown in blue) learns to classify the images in the dataset. The classifier first outputs a class activation map (CAM); then, obtains CAM's local maximas using a peak stimulation layer (PSL). To train the classifier, the classification loss is computed using the average of these local maximas. As the CAM peaks represent located objects, we select a proposal for each of these objects to obtain pseudo masks. The second component (shown in green) trains a Mask R-CNN on these pseudo masks.

segmenting C classes of objects. For a training image, the image-level label is given as $Y = [y_1, y_2, ..., y_C]$, where $y_i = 1$ or 0, indicating whether the image has an object of class i. We describe our components in more detail below, and also investigate a post-processing steps that can improve Mask R-CNN's final output.

4.3.1 Pseudo Mask Generation Branch

We rely on PRM [141] to generate segmentation seeds that identify salient parts of the objects. These seeds help in generating pseudo masks as a source of supervision for Mask R-CNN. Following PRM's methodology, we train a CAM-based classifier which has a fully convolutional network (FCN) followed by a peak stimulation layer (PSL). As shown in Figure 4.2, the FCN outputs a class activation map (CAM) which specifies the class confidence at each image location. Then, PSL outputs N^c local maxima of CAM within a window size r, namely, $L^c = \{(i_1, j_1), (i_2, j_2), ..., (i_{N^c}, j_{N^c})\}$ which represents locations in the CAM for the c-th object class (more details in Zhou et al. [141]). In order to boost the activations of these local maxima, their average activation is first computed as, $s^c = \frac{1}{N^c} \sum_{(i_k, j_k) \in L^c} M^c_{i_k, j_k}$, where M^c is the activation map corresponding to class c. This average is then used for binary classification, specifically the multi-label soft-margin loss [64], which for a training example with label y is computed

Algorithm 1: WISE-ILS training

Train a CAM-based classifier C until convergence as in PRM [141];
while $iter < max\_iter$ **do**
    Randomly sample a training image I;
    Generate a set of proposals P for I;
    Use PSL on C to obtain the set of peaks L for I;
    Initialize an empty list of Targets T;
    for $(i_k, j_k) \in L$ **do**
        Select a proposal (G_k, b_k) randomly using Eq. 4.2, it has to intersect with (i_k, j_k);
        Add G_k to list T;
    end for
    Compute $\mathcal{L}(I, T, \theta)$ as in Eq. 4.3;
    Update the weights for θ using back-propagation;
end while

as

$$M(s^c, y) = \log(1 + \exp(-s^c \cdot y)). \tag{4.1}$$

This classification component is trained until convergence. We then generate the pseudo masks for the training images by using the trained classifier and an off-the-shelf object proposal method (specified as the dotted line in Figure 4.2). The peaks obtained from PSL, which represent object locations in the image, are replaced with proposal masks based on their "objectness", which are scores given by the proposal method as confidence measure for being objects. We adopt a denoising strategy where we select a proposal randomly based on its objectness score: proposals with higher objectness are more likely to be selected. More formally, to obtain the mask for an object located at peak (i, j), we first generate a set of n proposals whose masks intersect with (i, j), namely, $P = \{(G_1, b_1), (G_2, b_2), ..., (G_n, b_n)\}$ with mask G_k and objectness score b_k. Then, the probability of selecting a proposal mask G_k is

$$P(G_k) = \frac{b_k}{\sum_{j=1}^{n} b_j}. \tag{4.2}$$

The rationale behind selecting proposals randomly is that they have common pixels that correspond to the salient parts of the located object, despite the fact that they have different objectness. While proposal masks are not originally associated with a class label, we get the object class label information from CAM and assign it to the corresponding proposals. These proposals can be used as pseudo masks to train a fully supervised instance segmentation method.

Figure 4.3: **Inference.** At test time, only the trained Mask-RCNN is required to output the prediction masks in the image. As an optional refinement step, the predicted masks can be replaced with the object proposals of highest Jaccard similarity.

4.3.2 Fully Supervised Segmentation Branch

We can construct the segmentation labels for all the training images by using the trained pseudo mask generation branch. These are used as supervision to train a Mask R-CNN (shown as green components in Figure 4.2). Depending on the application, other choices of fully supervised methods can be used instead of Mask R-CNN: if the goal is to perform instance segmentation at real-time, one can consider training a YOLACT [13], and for semantic segmentation, one can consider training a DeepLab [19] segmentation network.

Mask R-CNN [49] combines Faster R-CNN [103] and FCN-based [79] methods to first detect the objects and then segment them. For an image I, with target pseudo masks T, Mask R-CNN with parameters θ is trained by optimizing the following objective function,

$$\mathcal{L}(I, T, \theta) = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}}, \tag{4.3}$$

where $\mathcal{L}_{\text{cls}}$ is the classification loss for the detected objects, $\mathcal{L}_{\text{box}}$ is the localization loss for the detected objects, $\mathcal{L}_{\text{mask}}$ is their segmentation loss. These terms are explained in more detail in the original Mask R-CNN paper [49].

At test time, we can simply use the trained Mask R-CNN to predict the object masks for an unseen image. To refine these masks, we leverage the same object proposal method as that used in training. In turn, we replace each predicted object mask with the proposal of highest Jaccard similarity. Figure 4.3 illustrates how this refinement process can lead to a better object mask.

Method	Supervision	mAP25	mAP50	mAP75	ABO
Mask R-CNN [49]	pixel-level	58.9	51.4	32.4	-
DeepMask [57]	pixel-level	-	41.7	09.7	-
PRM [141]	image-level	44.3	26.8	09.0	37.6
PRM+Density [24]	image-level++	48.5	30.2	14.4	44.3
DeepMask [57]	bounding box	39.4	08.1	-	-
WISE-ILS (Ours)	image-level	48.5	40.4	22.2	51.3
WISE-ILS+Refine (Ours)	image-level	**49.2**	**41.7**	**23.7**	**55.2**

Table 4.1: **PASCAL VOC 2012.** Comparison of our framework (WISE-ILS) against other methods on various levels of supervision. WISE-ILS+Refine uses the refinement step shown in Figure 4.3. Mask R-CNN and DeepMask use full supervision, whereas PRM uses image-level labels. Similar to WISE-ILS, PRM and PRM+Density leverage a pretrained proposal method. Requiring stronger supervision than WISE-ILS, DeepMask and PRM+Density have access to bounding box and image-level counts, respectively.

4.4 Experiments

In this section, we demonstrate the efficacy of our method by comparing it against previous methods and analyzing the pseudo masks.

4.4.1 Experimental Setup

We follow the setup by Zhou et al. [141], Cholakkal et al. [24] for a fair benchmark, where the model only has access to an off-the-shelf proposal method and image-level labels for the training set. Also from their setup, we adopt the evaluation metric, mean average precision for Intersection-over-Union (IoU) of 0.25, 0.5, and 0.75.

Like other works in the literature of weakly supervised methods, we perform all comparisons on the PASCAL VOC 2012 dataset [35]. The dataset represents a diverse set of everyday scenes. It is divided into 1442 images for training, and 1449 images for validation. Annotators for this dataset acquired per-pixel labels for 20 objects, ranging from inanimate objects such as airplanes and bikes, and living objects such as humans and horses. However, we only use the image-level labels to train our methods.

Method	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
Mask R-CNN	71.2	0.3	72.2	53.2	29.8	68.7	47.3	77.1	13.3	54.7	41.0	65.5	51.5	69.6	57.8	31.0	46.9	45.6	69.7	61.4	51.4
WISE-ILS	59.2	0.6	62.6	38.6	18.8	57.3	31.7	66.9	8.3	40.5	11.0	55.5	48.7	60.2	34.4	24.4	38.3	33.1	61.7	56.9	40.4
WISE-ILS+Refine	63.2	0.3	60.7	39.1	21.0	59.4	31.9	68.6	9.2	43.1	15.6	58.0	48.6	62.3	36.4	21.9	38.8	34.3	65.5	56.9	41.7

Table 4.2: **PASCAL VOC 2012.** Per-class comparison against the mAP_{50} metric on PASCAL VOC 2012 validation set. Mask R-CNN was trained with the ground-truth per-pixel labels.

4.4.2 Implementation Details

We discuss our method's procedure and parameters below. We also plan to make the code publicly available.

Network architecture. As a common practice, we use the ResNet-50 [48] that is pretrained on ImageNet [31] as the backbone for PRM and Mask R-CNN. Unlike PRM, Mask R-CNN's backbone is equipped with a feature pyramid network [76] that extracts features at different resolutions. The pretrained weights, along with the rest of the parameters, are then finetuned on the PASCAL VOC 2012 training set. The remaining parameters of PRM and Mask R-CNN are in the implementation details discussed in Zhou et al. [141], and He et al. [49], respectively. We used a pretrained SharpMask [93] for our proposal method.

Optimization parameters. Following the official code of Mask R-CNN, we scale its input images so that the short axis has a minimum of 800px and the long axis a maximum of 1333px. Using a single GPU of TitanX, we set our batch size as 1 and train using the SGD optimizer with a learning rate of 0.00125 for 50K iterations. This learning rate was adjusted from He et al. [49], where they used a bigger batch size. We also augment the dataset with horizontal flips and color jittering as recommended by Deng et al. [31]. PRM was trained as described in Zhou et al. [141].

4.4.3 Comparison to Previous Work

We first quantitatively compare our approach against previous methods that use the same supervision as ours; that is, image-level labels, an object proposal method, and a ResNet-50 backbone pretrained on ImageNet. Table 4.1 summarizes the results on the PASCAL VOC 2012 dataset. Our method significantly outperforms the current state-of-the-art by a large margin with respect to Average Best Overlap (ABO) [95], mAP25, mAP50, and mAP75. Further, WISE-ILS without refinement also beats current state-of-the-art. Even more so, our method outperforms Cholakkal et al. [24] which uses slightly stronger labels than image-level. Their labels distinguish between images with 0, 1, 2-4, and 4-or more objects.

Metric	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	motor	person	plant	sheep	sofa	train	tv	Avg.
Mask R-CNN + GT	92.4	15.1	97.4	87.9	91.4	94.4	93.8	100	68.2	93.4	88.8	97.4	96.4	95.3	92.8	89.3	92.3	97.7	100	100	89.2
Pseudo Masks	24.5	1.0	29.1	18.7	11.3	38.6	26.6	43.1	8.0	35.6	6.1	38.8	46.2	23.8	10.7	7.4	35.9	29.4	41.6	39.1	25.8
WISE-ILS	43.8	3.2	43.8	35.9	16.8	51.9	36.3	56.8	7.3	45.8	15.1	53.5	59.8	45.5	18.2	10.9	47.3	38.9	61.5	58.5	37.5

Table 4.3: **PASCAL VOC 2012 training set.** Comparison of the generated pseudo masks, and WISE-ILS's predicted masks with respect to mAP50. WISE-ILS was trained on a set of pseudo masks, and was able to output better masks for the same training images. Mask R-CNN + GT was trained on the ground-truth per-pixel labels.

Figure 4.4 visualizes qualitative results of WISE-ILS for each category. We further report the per-class results in Table 4.2 and compare it against Mask R-CNN trained on the true masks with respect to mAP50. This illustrates that our results are competitive against fully-supervised methods.

Our method can also compete with those that use stronger supervision. Against DeepMask [57], our method outperforms two of their methods, one that uses bounding boxes as labels, and the other that uses full supervision as labels (see Table 4.1). Compared to Mask R-CNN trained on the pixel-level labels, our method still has a large room for improvement, which can be bridged by either improving the object localization component or the object proposal method.

The overall results suggests that Mask R-CNN can effectively train from noisy, incomplete labels. The labels are noisy because the proposal masks are not perfect, and incomplete because PRM does not locate all the objects in the image. Indeed, we hypothesize that using a better object localizer such as that of Cholakkal et al. [24] would lead to better results. But we leave that for future work.

Analysis of Pseudo masks

We measure the generated pseudo masks performance by computing the mAP50 between the ground-truth and the generated masks. We also compute the mean absolute error to determine the number of identified objects in the images. These results are summarized in Table 4.3, which show that a large room for improvement is required for both metrics. Examples of the synthesized masks are shown in Figure 4.1, where one can see that the pseudo masks are not of high quality, yet the trained Mask R-CNN is able to output good masks in Figure 4.4.

Ablation Studies

The object sizes and the number of objects in an image can have severe impact on the performance of an instance segmentation model. Figure 4.5 shows that WISE-

Figure 4.4: **Qualitative results.** Qualitative results of WISE-ILS on PASCAL VOC 2012 val. set. The images illustrate the predicted masks of the trained Mask R-CNN for different classes.

ILS struggles with segmenting small objects, and when the number of objects is larger than 4. In fact, there is a heavy decline in performance when the number of objects is more than 1. More robust than WISE-ILS, a Mask R-CNN trained on per-pixel labels is able to maintain higher performance with small objects and with images with larger number of objects. In addition, such Mask R-CNN performs significantly better than WISE-ILS for small objects. This suggests that the pseudo masks trained by WISE-ILS are likely far from accurate.

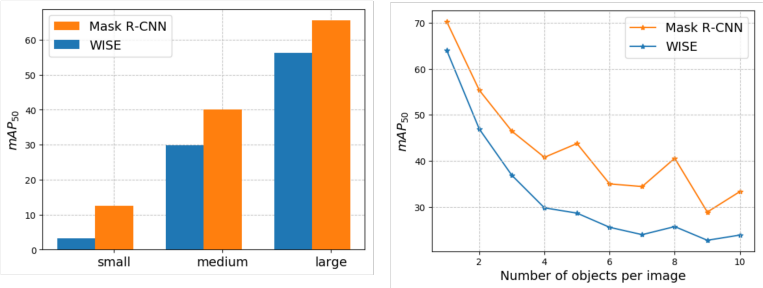


Figure 4.5: **Statistical Analysis.** The left figure illustrates the performance of WISE-ILS and a Mask R-CNN trained on per-pixel labels across various object sizes; and the right figure illustrates the same benchmark but across images with different number of objects.

4.5 Limitations

The performance of our method and the competing methods [141, 24] in this problem setup are heavily influenced by the quality of the proposal network. The proposal network used for these methods was trained on class-agnostic masks of the PASCAL [35] dataset. Thus, we plan to decrease the amount of supervision by having a proposal network only trained on object classes that do not exist on the training set.

4.6 Conclusion

We proposed a weakly supervised instance segmentation method that follows a two-stage pipeline for training on image-level labels. In the first stage, it uses class activation maps with a peak stimulation layer to locate the objects in the training images, and then object proposals to generate pseudo masks for these objects. In the second stage, we use Mask R-CNN to train on the pseudo masks in a fully supervised manner. We evaluate on PASCAL VOC 2012, a standard benchmark for weakly supervised methods, where Mask R-CNN trained on pseudo masks outperformed not only methods with the same level of supervision, image-level labels, but also methods that use counts and bounding boxes in their supervision.

Chapter 5

Object Localization for Dense Scenes with Count Supervision

We propose a weakly supervised localization model (WSLM) that can localize objects in congested scenes and still perform accurate count estimation. Unlike most existing counting methods, WSLM only requires count supervision, instead of point-level annotations. WSLM follows 3 main steps. First, it trains an extension of Glance [16], which we call as Glance-ram. to convergence. Glance-ram can output an additional regression activation map (RAM). Second, it uses RAM to score and select a set of region proposals for the training images and converts them to pseudo point-level annotations. These proposals are generated using a pretrained region proposal network (RPN). Third, it trains a state-of-the-art localization method like LCFCN (Chapter 2) on these pseudo point-level annotations. WSLM is flexible in that other counting methods with an attention mechanism can be considered for the first step, other proposal methods for the second step, and other localization networks for the third step. We evaluate WSLM on 6 datasets of congested scenes, including ShanghaiTech, UCSD, and Mall. With count supervision, we deliver state-of-the-art performance with Glance-ram, and a first strong baseline for object localization in congested scenes with WSLM.

5.1 Introduction

Object *counting* and *localization* are important tasks for understanding highly congested scenes. Object counting is the task of counting objects of interest, whereas object localization is the task of locating these objects in the image. These methods provide promising solutions for applications such as public safety, crowd monitoring, and traffic management. For congested scenes, some methods can only output the count [16]; however, for many applications, getting only the count is not enough. These applications demand localization as well, which could be critical for making decisions in high-risk environments like riots. Further, localization can help users understand the insights of the counting method to judge whether it works in practice. The last row in Figure 5.1 shows examples of objects being localized

Figure 5.1: **WSLM overview.** The weakly supervised localization model (WSLM) learns to count and localize objects in crowded images. It learns to transform image-level counts into localization pseudo ground truth points and trains a counting and localization network on them.

in congested scenes.

Despite being essential in a myriad of applications however, both tasks of object counting and localization remain unsolved. Their challenge in practical scenarios arises from the state of real-life images, in which objects are often severely occluded, vary in scale, illumination and clutter, or are distributed unevenly within the scene. Existing methods for object counting and localization are either density-based or localization-based. These methods are explained in detail in Chapter 2.

In this work, we are interested in a model that is able to count and localize objects in a scene, but that is trained only on count data. This would alleviate the burden of labeling each image with points placed on every object, which we argue,

constitutes a more costly endeavor.

At first glance, it would seem that the effort required for collecting point-level annotations and image-level counts are equivalent — after all, a human annotator needs to point to every object in the image in order to count them. But in many cases, acquiring object counts in images requires much less human effort than annotating the location of each object. For example, according to subitizing theory, when training images contain 4 or less objects, the annotator can obtain the object count much faster than with point annotations [16]. Another example is the case of labeling a large sequence of image frames: instead of counting and locating the objects in each frame separately, an annotator can do count and annotate the objects only in the first frame, and then adjust her count and annotations by adding or subtracting objects as they appear or disappear in subsequent frames. This latter process takes roughly the same amount of time as collecting point-level annotations for one frame, thus vastly decreasing the total labeling time for the entire sequence of frames. A final example is the case of registration-based systems, where object counts can be obtained for free: if a manager orders a certain number of cans to be placed in a glass-windowed fridge, then the images taken of that fridge are already labeled with the count of cans.

Many methods exist for weakly supervised localization, but they are not suitable for highly congested scenes. Examples include CAM-based models [109, 1], which localize objects for datasets with image-level supervision. These models treat the problem as a classification task, as a consequence of which they have two problems. First, they can only localize one object per image and fail if the image contains more than one object of the same class. Second, they only work for multiclassification problems and fail if the dataset contains only one class, because the classification model relies on the background images in order to identify the objects of interest. PRM [141] and C-WSL [40] are two weakly supervised localization algorithms that extend basic CAM-based methods to localize more than one object per image, but they rely on having few objects per scene and on many background images. Glance [16], on the other hand, can learn to count objects in dense scenes using counts only, but does not perform object localization.

Therefore, we propose our Weakly Supervised Localization Model (*WSLM*) that can localize objects in congested scenes and still perform accurate count estimation. Unlike most existing counting methods that require point annotations, WSLM only requires count supervision. WSLM follows three main steps. First, it trains to convergence the *Glance-ram* model, which is a model that extends Glance [16] by adding a Regression Activation Map (RAM). Although similar to class activation maps, RAM highlights the discriminative regions that the model uses to build its regression output. Further, instead of global average pooling like in most CAM methods, Glance-ram uses local average pooling to have peaks emerge

at different locations of the RAM. These peaks are meant to represent the objects of interest. Glance-ram has close similarities to PRM [141], which uses peaks to get object instances. The main difference between the two is that PRM uses the classification loss, and ours uses the regression loss. As a side note, incorporating RAM and local average pooling to Glance has improved its counting ability by a good margin.

In the second step of WSLM, proposals are first generated for each training image using a pretrained region proposal network (RPN). Then, each proposal is scored based on three metrics. The first metric is the *objectness score* of a proposal, which indicates how likely it is that an object is a proposal, and which is provided by the proposal network. The second metric is how much the proposal overlaps with activated regions in the generated RAM. The third metric is based on generating peak response maps (PRMs) as in Zhou et al. [141]. In this case, the location of the proposal centroid is backgpropagated with respect to the generated RAM, resulting in a PRM for that proposal. Each proposal is then ranked by aggregating their metric scores, and the top proposals are selected using the count-based region selection method in Gao et al. [40]. Finally, the selected proposals are converted to point-level annotations by taking their centroids.

In the final step of WSLM, a state-of-the-art localization method like LCFCN (Chapter 2) is trained on these pseudo point-level annotations. At test time, only the trained LCFCN is kept while the rest of the components are discarded. This trained LCFCN performs both counting and localization.

Since no direct relevant work exists for this particular setup, we compare our methods Glance-ram and WSLM against Glance [16] and the fully supervised LCFCN [65]. We benchmark our methods against various counting datasets such as Trancos [45], Shanghai Tech B [136], Park Lot [29], Penguins [8], UCSD [14], and Mall [17]. Based on these results, Glance-ram achieves better count accuracies than Glance, while WSLM achieves better results than Glance in most cases, while also obtaining good localization performance.

We summarize our contributions as follows: we (1) present WSLM, a novel framework that can count and locate objects with count-level supervision for dense scenes; (2) present Glance-ram, which extends Glance with an attention map that achieves state-of-the-art for counting objects with count supervision; and (3) show that WSLM achieves better count accuracy than Glance while still being able to localize objects. While WLSM does not perform as well as the the fully supervised LCFCN, it serves as a strong baseline and hopefully inspires further research in this challenging yet interesting problem setup.

5.2 Related Work

Our work lies at the intersection of three main topics: object counting and localization, multiple-instance learning methods for weak supervision, and object proposal methods. In the remainder of this section, we briefly discuss these topics.

Counting and Localization. Methods under this topic can be grouped into two main categories: (1) regression-based methods, and (2) detection-based methods.

Regression-based methods optimize the localization-based counting loss objective. Models in this category are Glance [16] and density-based methods [68, 45, 136, 89, 71, 24]. Density-based methods require point-level annotations that makes them unsuitable for the problem setup of this work. On the other hand, Glance only requires count-level supervision, but only outputs the count rather than the object locations, which is an important task that we consider in this work.

Detection-based methods learn to find the object locations, but also their width and height [103, 100, 78]. Even though perfect detection implies perfect counting, these methods fell out of fashion in favor of regression-based approaches for the counting task. Since detection is a harder task than counting, learning to count through detection can lead to worse counting results than explicitly learning to count. A further limitation is that these methods require bounding box annotations, which are costly to acquire [10]. An easier task than detection is localization where objects only need to be located rather than have their heights and widths estimated. One example of an effective localization-based method is LCFCN (Chapter 2), which is state-of-the-art for locating objects of interest. The reasoning behind why LCFCN works well for counting is counting objects requires locating the objects in the image as a perquisite.

Multiple-Instance Learning Methods for Weak Supervision. Many weakly supervised localization methods use multiple-instance learning (MIL) [33]. MIL is a weakly supervised learning setup, whereby instead of receiving individually-labeled instances as input, the model receives a set of labeled bags of instances. A bag is labeled positive if at least one of its instances is positive, and negative otherwise. Li et al. [69] study the problem of weakly supervised object localization where image-level annotations are available. They present a domain adaptation approach that has two steps. In the first step, they adapt the classification by using a mask-out strategy to filter the noisy object proposals; in a second step, they adapt detection by learning a Faster RCNN [103] using bags of instances. This approach is for weak supervision when image annotations are available, and is not directly comparable to our method as it use a classification model rather than a regression. Tang et al. [119] convert the weakly supervised detection problem into a MIL problem where object detectors are hidden nodes in the network. They integrate a

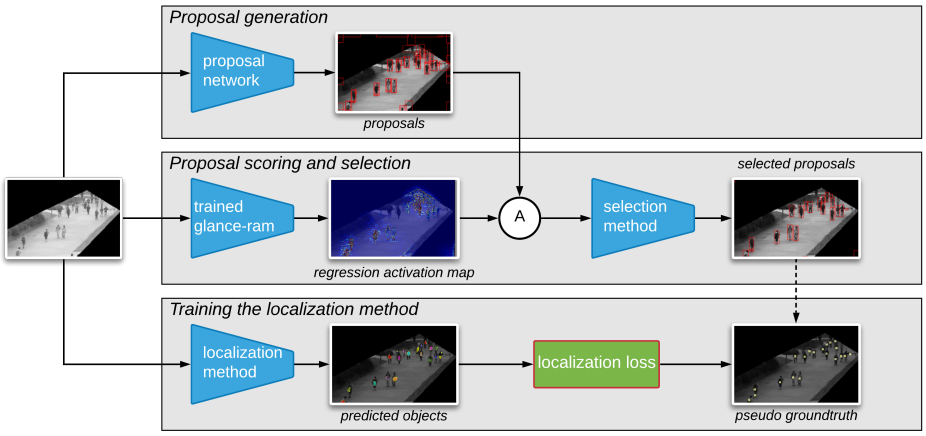


Figure 5.2: **Training WSLM.** WSLM consists of 4 components: (1) a proposal generation network that generates agnostic object proposals, (2) a trained glance-ram that scores the proposals based on how much they overlap with the regression activation map, (3) a proposal selection method that chooses the top C proposals based on their score and, (4) a localization and counting network that is trained on the pseudo ground truth generated by the proposal selection method.

multiple-instance detection network and instance classifiers into a single network to locate the most discriminative regions of an image. C-MIL [126] introduces a continuation optimization method into MIL to prevent the MIL setting from getting stuck in local minima. C-WSL [40] is the only work on weakly supervised localization with count supervision. The approach is based on ranking region proposals to get high-quality proposals. However, it relies on a classification network that requires many background images, which is not the case for most counting datasets.

Object proposals. Many weakly supervised methods rely on object proposals [51] to help with the task of detection [120, 12] and segmentation [90, 10, 141, 60]. Object proposals are class-agnostic methods that can output thousands of object candidates per image and have seen great progress over the last decade [124, 143, 4, 82, 91, 92]. Region Proposal Networks (RPN) [84] and SharpMask [92] are popular deep-learning based object proposal methods that have been successfully applied to many weakly supervised computer vision problems. However, their output object masks cannot be directly used for localization as they belong to arbitrary objects and have no class labels.

5.3 Proposed Method: WSLM

The goal of WSLM is to localize objects in congested scenes and still perform accurate count estimation. It does this with the help of an attention-based counting method, a proposal network, and a fully supervised localization method. Roughly, WSLM performs the following steps in sequence.

1. Train the attention-based counting method on the count labels. This generates an activation map that scores regions in the image based on how likely it is that they contain the objects of interest.

2. Use the generated activation maps to select among the set of proposals generated by a region proposal network (RPN). The centroids of the selected proposals are then set to be the pseudo ground-truth point-level annotations.

3. Use these annotations to train a fully supervised localization method (such as LCFCN).

We explain these steps and WSLM's components in more detail below.

5.3.1 Attention-Based Counting

We propose Glance-ram, an attention-based counting method inspired by Glance [16] and class activation maps (CAM). The goal of Glance-ram is to generate regression activation maps (RAM), which represent discriminative regions that help the model obtain the regression output.

Glance-ram differs from Glance in that it has no fully connected layers. The output of the last convolutional layer is used as the regression activation map to visualize the regions of interest. In order to aggregate the output of the activation map, we use local average pooling (LAP) that is inspired by Zhou et al. [141]. We do not use global average pooling (GAP) because it assigns equal importance to all responses, which makes it hard to distinguish between the foreground from the background.

LAP works as follows. First, a set of K local maxima L are obtained from RAM within a window of size r. $L = \{(i_1, j_1), (i_2, j_2), ..., (i_k, j_k)\}$, which represents locations in the regression activation map. In order to boost the activations of these local maxima, their average activation is first computed as $s = \frac{1}{K} \sum_{(i_k,j_k) \in L} M_{i_k,j_k}$, where M is the regression activation map. In order to stimulate only discriminative local maxima, we select the maxima that are higher than the median across the maxima and compute the mean of the selected maxima.

The regression output is the mean across the selected local maxima. This mean value is used to compute the mean squared error with respect to the provided count

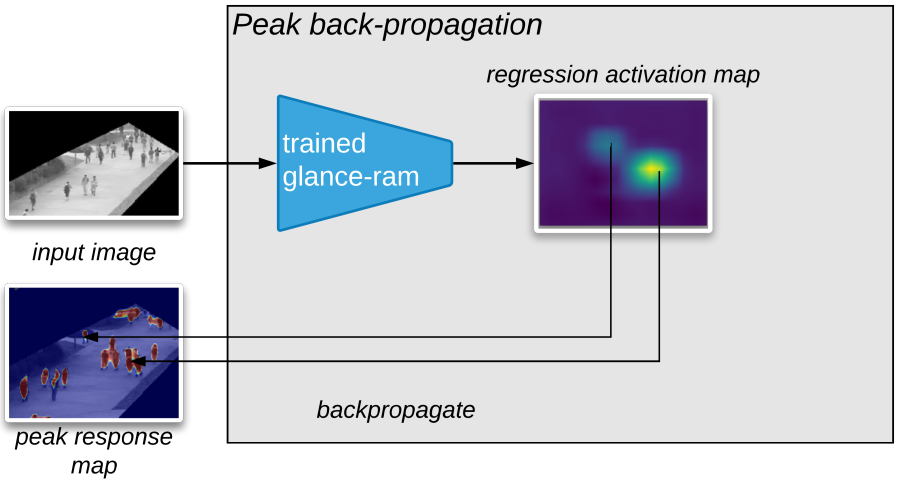


Figure 5.3: **Glance-ram.** Given the regression activation map as M, Glance-ram generates a PRM for a spatial location j as follows. First set the gradient to 1 for M_j and 0 for the rest of the activations in M. Then, backpropagate the gradient signal to the input image to obtain the informative regions for spatial location j. For a specific proposal i, we generate its PRM using the proposal's centroid as its spatial location.

labels; this error is then is backpropagated to update the model. During training, the described forward pass and backward pass are repeated until Glance-ram converges.

5.3.2 Generating Point Annotations

The next step is to generate pseudo point-level annotations for the training images. To do so, we leverage the trained regression model, Glance-ram, and an off-the-shelf object proposal method by following these steps. First generate a set of n proposals using the object proposal method. Then, score each proposal i using three metrics. The first metric is the objectness score of the proposal, denoted as O_i, which is given by the proposal method. This score is class-agnostic in that it does not indicate whether the proposal belongs to an object of interest. Fortunately, metric two and metric three address this limitation. Metric two uses the regression activation map (RAM) generated from the trained Glance-ram. Let Z be the set of pixel coordinates within a proposal i, M be the regression activation map, and M_z

be the RAM score at pixel z. Then the score for proposal i is the average

$$\frac{1}{|Z|}\sum_{z\in Z} M_z.$$

This represents how much the proposal overlaps with what Glance-ram sees as important regions for getting the object count. The third metric is based on the Peak Back-propagation method in Section 3.3 in Zhou et al. [141]. For each spatial location, this method can generate a fine-detailed instance-aware representation known as the peak response map (PRM). Given M as the regression activation map, we can generate a PRM for spatial location j as follows. First set the gradient to 1 for M_j and 0 for the rest of the activations in M. Then, backpropagate the gradient signal to the input image to obtain the informative regions for spatial location j. For a specific proposal i, we generate its PRM using the proposal's centroid as its spatial location. For simplicity, denote R_i as the PRM for proposal i. Then, the score of proposal i with metric three is the average,

$$\frac{1}{|Z|}\sum_{z\in Z} R_z,$$

where R_z is the proposal's PRM score at pixel z, and Z is the set of pixel coordinates in proposal i. Putting these three metrics together, the score of a proposal i is

$$\sum_{z\in Z_i} \alpha M_z + \beta R_z + \gamma O_z, \tag{5.1}$$

where α, β, and γ are coefficients for weighing between the different metrics.

The next step in generating the pseudo point-level annotations is to select the best proposals for each training image. Given the score of each proposal, we select the best k proposals using CRS [40], a count-based region selection method, where k is the provided object count annotation for a given image. In contrast to naively selecting the top k proposals, CSR attempts to select k distinct proposals, each covering a single object. Finally, the centroids of the selected proposals are used as the pseudo point-level annotations.

5.3.3 Training the Localization Network

The goal of the localization network is to count and localize the objects in the image. The network is directly trained on the generated point-level annotations in Section 5.3.2. We use LCFCN [65] instead of the more commonly used density-based localization methods for the following reason. LCFCN does not require

Dataset	class	# images	# Resolution	# Min	# Ave	# Max	# Total count
UCSD [14]	person	2,000	158x238	11	25	46	49,885
Trancos [45]	car	1,641	640x480	9	36.54	95	46,796
Mall [17]	person	2,000	320x240	13	-	53	62,325
PKLot [29]	person	2,000	320x240	13	-	53	62,325
Shanghai [136]	person	716	768x1024	9	123	578	88,488
Penguins [8]	penguin	80,095	2048x1536	0	7.18	67	575,082

Table 5.1: **Datasets' statistics.**

obtaining density maps, which need domain knowledge about the size of the object [71]. At test time, we only use the trained LCFCN to predict the locations of the objects in the image. Thus, other components of WSLM can be discarded at deployment.

5.4 Experiments

Our goal is to show that WSLM can (1) effectively localize objects in dense scenes, and at the same time (2) get the right object count. To evaluate the localization capabilities of WSLM, we compare it against the fully supervised localization method LCFCN. To evaluate its object count capabilites, we compare WSLM against Glance, a competitive method for counting objects that only uses object count as supervision.

5.4.1 Experimental Setup

Datasets. We experiment on a variety of challenging counting datasets. They range from low to high density scenes and pose different challenges regarding object scale variation, clutter, and occlusion. Table 5.1 displays the statistics and brief overview of the datasets used in our benchmark. Originally, these datasets are labeled with point-level annotations; that is, each object has a single pixel labeled with the object class. However, we only consider the count-level supervision for training WSLM.

Evaluation Metrics. We evaluate WSLM in terms of count and localization estimation. We report the count estimation error using the mean absolute error (MAE):

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{5.2}$$

where N is the number of test samples, y_i is the ground truth count and $\hat{y}_i$ is the estimated count corresponding to sample i.

For the localization estimation error, we consider the grid average mean absolute error (GAME) [45]. It is computed as

$$GAME(L) = \frac{1}{N}\sum_{i=1}^{N}(\sum_{l=1}^{4^L}|D_i^l - \hat{D}_i^l|) \tag{5.3}$$

where D_i^l is the number of point-level annotations in region l, and $\hat{D}_i^l$ is the model's predicted count for region l. $GAME(L)$ first divides the image into a grid of 4^L non-overlapping regions, and then computes the sum of the MAE across these regions. Note that $GAME(0)$ is equivalent to MAE.

Methods Definition. We compare the following methods.

Glance: uses the ImageNet pre-trained network as described in Chattopadhyay et al. [16].

Glance-ram: uses the ImageNet pre-trained ResNet50 network. It also has a regression attention map followed by a local average pooling operations.

WSLM: first scores each proposal using Eq. 5.1. It then selects the best proposals to get the pseudo point level annotations for the training images. Finally, it trains an LCFCN model on those pseudo annotations.

LCFCN: is the ResNet50 variation of the LCFCN method described in Laradji et al. [65].

Implementation Details. We optimize all our models using ADAM [58] with learning rate of 1e-5 and weight decay of 0.0005. The proposals are generated using a pretrained region proposal network (RPN) [103]. We use a threshold of 0.1 for the CSR proposal selection method, and set the weight coefficients α, β, and γ as $\frac{1}{3}$.

Note that the pretrained region proposal network (RPN) [103] generates 1000 proposals per image. We use the same pretrained RPN for all our datasets. The RPN used here consists of a ResNet-50 [48] as backbone and a feature-pyramid network [76] to extract features for different resolutions.

5.4.2 Experimental Results

Below we describe our benchmark datasets and the results of our methods. The results include quantitative analysis on the counting and localization performance, and qualitative illustrations of the predictions of our methods on test images.

Method	MAE	GAME
Glance	4.20	-
Glance-ram (ours)	3.23	-
WSLM (ours)	2.82	18.20
LCFCN	0.99	4.12

Table 5.2: Results on the UCSD Dataset.

UCSD [14]. Perhaps the first dataset for counting people, UCSD consists of images collected from a video camera at a pedestrian walkway. This dataset has less object density compared to other counting datasets; but it is still challenging due to pedestrians overlapping each other, which makes counting and localization difficult.

Each frame in UCSD is 238x158 pixels, which is not suitable for our ResNet based models. Thus, we resize the frames to 952x632 pixels using bilinear interpolation. This practice has also been adopted in Li et al. [71]. Additionally, as common practice [14], we use the frames 601-1400 as training set and the rest as test set.

Table 5.2 shows that Glance-ram outperforms Glance, suggesting that having an attention map is an important component. On the other hand, WSLM outperforms both Glance and Glance-ram with respect to MAE; but it performs poorly with respect to GAME against the fully supervised method. The most challenging aspect of this dataset, however, is in the cluttered regions, where the pseudo point-level annotations are not on top of each pedestrian (Figure 5.4). Further, a pseudo point-level annotation is selected on the top left corner of the image, although it has no pedestrian (Figure 5.4). This limitation is likely due to overfitting. Despite the challenges, Figure 5.4 shows that WSLM achieves comparably good localization by finding the right objects of interest.

Trancos [45]. This dataset consists of images taken from traffic surveillance cameras for different roads. The task is to count the vehicles present in the regions of interest. These vehicles often highly overlap in the image [45], making the dataset challenging for localization. While each vehicle is originally labeled with a point annotation, we only use the vehicle count as training labels.

The results shown in Table 5.3 indicate that Glance-ram achieves lower MAE than Glance. They also indicate that WSLM performs better in object counting, yet it can perform good localization. Compared to the fully supervised LCFCN, WSLM performs poorly mainly due to the quality of the pseudo point-level annotations.

Method	GAME0	GAME1	GAME2	GAME3
Glance	8.70	-	-	-
Glance-ram (ours)	6.92	-	-	-
WSLM (ours)	6.70	12.90	17.60	24.00
LCFCN	3.57	4.98	7.42	11.67

Table 5.3: Results on the Trancos Dataset.

Mall [17]. Collected from a fixed camera installed in a shopping mall, Mall consists of 2000 frames of size 320x240. These frames have diverse illumination conditions and crowd densities, and the objects vary widely in size and appearance. Results are shown in Table 5.4.

Method	MAE	GAME
Glance	4.03	-
Glance-ram (ours)	3.38	-
WSLM (ours)	3.10	18.40
LCFCN	2.12	8.70

Table 5.4: Results on the Mall Dataset.

PKLot [29]. The dataset consists of images taken from a fixed camera at a parking lot in Curitiba, Brazil. The goal is to find the number and location of cars in the images. The original labels are bounding boxes over each car. For the ground truth point-level annotations, we pick the centroids of the bounding boxes, although we train our models using the number of cars only. We use the PUCPR subset of the dataset with the first half of the images as the training set, and the rest as the test set.

Method	MAE	GAME
Glance	1.82	-
Glance-ram (ours)	1.14	-
WSLM (ours)	1.23	2.50
LCFCN	0.21	0.23

Table 5.5: Results on the PKLot Dataset.

Results are shown in Table 5.5. The proposed Glance-ram outperforms both Glance and the proposed WSLM, while LCFCN outperforms all three methods.

However, WSLM has an advantage over both these methods as (1) it is trained on only count annotations and (2) it outputs localization (Figure 5.4). Note also that since WSLM is trained on pseudo point-level annotations, it is not surprising that it performs worse than LCFCN.

ShanghaiTech B [136]. Part B consists of images taken from the streets of metropolitan areas in Shanghai. Part B has 400 and 316 images, respectively. The dataset successfully creates a challenging image set with diverse scene types and varying density levels. However, the number of images for various density levels are not uniform, making the training and evaluation biased towards low density levels. Nevertheless, the complexities present in this dataset such as varying scales and perspective distortion has created new opportunities for more complex CNN network designs.

Method	**MAE**	**GAME**
Glance	23.50	-
Glance-ram (ours)	18.3	-
WSLM (ours)	38.70	82.30
LCFCN	13.14	23.30

Table 5.6: Results on the ShanghaiTech B Dataset.

The MAE, and GAME results are shown in Table 5.6. While Glance-ram outperforms Glance, WSLM achieves the worst result. This suggests that WSLM is not suitable for extremely dense scenes like ShanghaiTech. However, Figure 5.4 shows qualitative results for WSLM that indicate that it can perform some good degree of localization.

Penguins Dataset [8]. The dataset consists of images of penguin colonies that are collected from fixed cameras in Antarctica. We use the "mixed" dataset split where images in the training set come from the same cameras as those in the test set. Penguins can come in many different sizes: there are baby penguins, big penguins, small penguins, and so on.

Due to the size of the Penguins datasets, we report the results on the validation set of Penguins. Results are shown in Table 5.7 with qualitative examples in Figure 5.4.

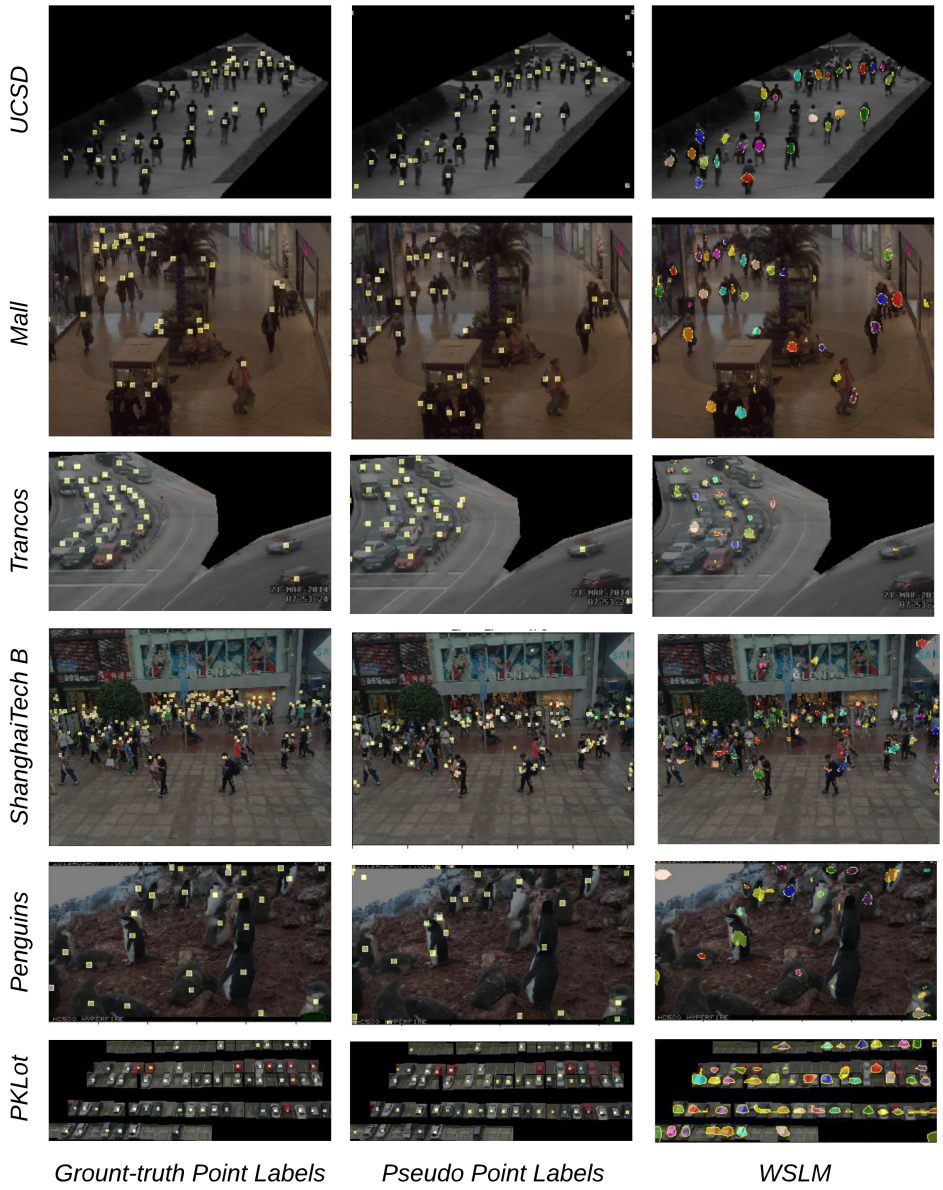


Figure 5.4: **Qualitative results.** Comparison between ground-truth point annotations, pseudo point annotations, and predictions made by WSLM.

Method	MAE	GAME
Glance	2.23	-
Glance-ram (ours)	2.6	-
WSLM (ours)	3.21	5.31
LCFCN	2.10	3.20

Table 5.7: Results on the Penguins Validation Set.

5.5 Limitations

Occlusions can affect the performance of our method. It is possible that the object count does not match the number of objects seen in the image because some objects might completely occluded by another object in front of it. In this case, it might be desirable to allow the model to localize less objects than the actual count during training.

The model might learn to localize the wrong object class due to ambiguity. The number of objects for one class might be similar to the class of interest. With only count supervision, it might be impossible for our model to learn which object class is of interest. Thus, it might be worth having at least few images with point-level supervision in order to identify the class for the objects of interest.

In contrast to Glance, our methods also require supervision from a pretrained proposal network that has been trained on a dataset like PASCAL [35] dataset. However, the proposal network has been trained on images that look very different than our datasets, and they were trained on class-agnostic bounding boxes.

5.6 Conclusion

We have proposed WSLM, a Weakly Supervised Localization Model that can localize objects in congested scenes and still perform accurate count estimation. WSLM performs three main steps. First, it trains Glance-ram to convergence, which extends Glance [16] by having a Regression Activation Map (RAM). Second, it uses RAM to score and select a set of region proposals for the training images and converts them to pseudo point-level annotations. We have shown the efficacy of WSLM on six datasets of congested scenes, including ShanghaiTech, UCSD, and Mall. With count supervision, we have delivered state-of-the-art performance with Glance-ram, and a first strong baseline for object localization in congested scenes with WSLM.

Chapter 6

Conclusion

Throughout this work, we have proposed four weakly supervised methods. They require annotations that are significantly cheaper to collect than the standard annotations. If the annotation budget is fixed, our weakly supervised methods achieve better results than the standard fully supervised methods. The first method can learn to count objects from point annotations only as opposed to detection-based methods that require bounding boxes. The second method can learn to segment objects by using point annotations as well. This significantly reduces the cost required for obtaining training labels for segmentation. In fact, it can take up to 1.5 hours to label a single image as opposed to around a half minute of annotation time. For the third method, we decreased the annotation required even further by proposing a segmentation model that only requires image-level labels. Finally, we proposed a detection model that only requires image-level counts. In many cases, image-level counts have similar cost to image-level labels and can come for free. Yet, counts provide more information than labels. Our benchmarks have shown that each of the proposed method achieve state-of-the-art results in their respective weakly supervised setup. Interestingly, in some cases our methods had comparable performance compared to fully supervised methods. One main weakness for our instance segmentation methods is that they require proposal methods that have been trained on large scale methods. Therefore, for future work, we plan instead to use an unsupervised methods for generating proposals. Another weakness for two-stage methods is error propagation. Errors from one-stage can affect the performance of the second stage. This can occur in WISE-ILS where the errors in pseudo label generation can heavily influence the training performance of Mask RCNN. For future work, we plan to extend these weakly supervised methods to work with synthetic data, to work in a few-shot setup [130], or to be used for active learning [110]. All these setups can significantly minimize the amount of annotation required.

Bibliography

[1] J. Ahn and S. Kwak. Learning pixel-level semantic affinity with image-level supervision for weakly supervised semantic segmentation. *CVPR*, 2018.

[2] J. Ahn, S. Cho, and S. Kwak. Weakly supervised learning of instance segmentation with inter-pixel relations. *arXiv*, 2019.

[3] L. Alvarez and L. Mazorra. Signal and image restoration using shock filters and anisotropic diffusion. *SIAM*, 1994.

[4] P. Arbeláez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. *CVPR*, 2014.

[5] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Learning to detect cells using non-overlapping extremal regions. *MICCAI*, 2012.

[6] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Learning to detect partially overlapping instances. *CVPR*, 2013.

[7] C. Arteta, V. Lempitsky, J. A. Noble, and A. Zisserman. Detecting overlapping instances in microscopy images using extremal region trees. *MIA*, 2016.

[8] C. Arteta, V. Lempitsky, and A. Zisserman. Counting in the wild. *ECCV*, 2016.

[9] M. Bai and R. Urtasun. Deep watershed transform for instance segmentation. *CVPR*, 2017.

[10] A. Bearman, O. Russakovsky, V. Ferrari, and L. Fei-Fei. Whats the point: Semantic segmentation with point supervision. *ECCV*, 2016.

[11] S. Beucher and F. Meyer. The morphological approach to segmentation: the watershed transformation. *MMIP*, 1992.

[12] H. Bilen and A. Vedaldi. Weakly supervised deep detection networks. *CVPR*, 2016.

[13] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: Real-time instance segmentation. *arXiv*, 2019.

[14] A. B. Chan, Z.-S. J. Liang, and N. Vasconcelos. Privacy preserving crowd monitoring: Counting people without people models or tracking. *CVPR*, 2008.

[15] R. M. Charles, K. M. Taylor, and J. H. Curry. Nonnegative matrix factorization applied to reordered pixels of single images based on patches to achieve structured nonnegative dictionaries. *arXiv*, 2015.

[16] P. Chattopadhyay, R. Vedantam, R. RS, D. Batra, and D. Parikh. Counting everyday objects in everyday scenes. *CVPR*, 2017.

[17] K. Chen, C. C. Loy, S. Gong, and T. Xiang. Feature mining for localised crowd counting. *BMVC*, 2012.

[18] L.-C. Chen, A. Hermans, G. Papandreou, F. Schroff, P. Wang, and H. Adam. Masklab: Instance segmentation by refining object detection with semantic and direction features. *CVPR*, 2018.

[19] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *PAMI*, 2018.

[20] X. Chen, R. Mottaghi, X. Liu, S. Fidler, R. Urtasun, and A. Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. *CVPR*, June 2014.

[21] Y.-T. Chen, X. Liu, and M.-H. Yang. Multi-instance object segmentation with occlusion handling. *CVPR*, 2015.

[22] J. Cheng and M. Lapata. Neural summarization by extracting sentences and words. *arXiv*, 2016.

[23] T. Ching, D. S. Himmelstein, B. K. Beaulieu-Jones, A. A. Kalinin, B. T. Do, G. P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M. M. Hoffman, et al. Opportunities and obstacles for deep learning in biology and medicine. *JRSI*, 2018.

[24] H. Cholakkal, G. Sun, F. S. Khan, and L. Shao. Object counting and instance segmentation with image-level supervision. *CVPR*, 2019.

[25] J. P. Cohen, G. Boucher, C. A. Glastonbury, H. Z. Lo, and Y. Bengio. Count-ception: Counting by Fully Convolutional Redundant Counting. *ICCV Workshops*, 2017.

[26] 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. *CVPR*, 2016.

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

[28] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. *CVPR*, 2005.

[29] P. R. De Almeida, L. S. Oliveira, A. S. Britto Jr, E. J. Silva Jr, and A. L. Koerich. Pklot–a robust dataset for parking lot classification. *ESA*, 2015.

[30] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the em algorithm. *JRST*, 1977.

[31] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. *CVPR*, 2009.

[32] Y. Deng, F. Bao, Y. Kong, Z. Ren, and Q. Dai. Deep direct reinforcement learning for financial signal representation and trading. *NNLS*, 2016.

[33] T. G. Dietterich, R. H. Lathrop, and T. Lozano-Pérez. Solving the multiple instance problem with axis-parallel rectangles. *AI*, 1997.

[34] N. Dvornik, J. Mairal, and C. Schmid. On the importance of visual context for data augmentation in scene understanding. *arXiv*, 2018.

[35] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010.

[36] A. Fathi, Z. Wojna, V. Rathod, P. Wang, H. O. Song, S. Guadarrama, and K. P. Murphy. Semantic instance segmentation via deep metric learning. *arXiv*, 2017.

[37] G. French, M. Fisher, M. Mackiewicz, and C. Needle. Convolutional neural networks for counting fish in fisheries surveillance video. *MVAB*, 2015.

[38] C.-Y. Fu, M. Shvets, and A. C. Berg. RetinaMask: Learning to predict masks improves state-of-the-art single-shot detection for free. *arXiv*, 2019.

[39] K. Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *BC*, 1980.

[40] M. Gao, A. Li, R. Yu, V. I. Morariu, and L. S. Davis. C-wsl: Count-guided weakly supervised localization. *ECCV*, 2018.

[41] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. *CVPR*, 2012.

[42] R. Girshick. Fast r-cnn. *ICCV*, 2015.

[43] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. *NIPS*, 2014.

[44] K. Gregor, I. Danihelka, A. Graves, D. J. Rezende, and D. Wierstra. Draw: A recurrent neural network for image generation. *arXiv*, 2015.

[45] R. Guerrero, B. Torre, R. Lopez, S. Maldonado, and D. Onoro. Extremely overlapping vehicle counting. *IbPRIA*, 2015.

[46] R. A. Haddad and A. N. Akansu. A class of fast gaussian binomial filters for speech and image processing. *SP*, 1991.

[47] B. Hariharan, P. Arbeláez, R. Girshick, and J. Malik. Simultaneous detection and segmentation. *ECCV*, 2014.

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

[49] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. *ICCV*, 2017.

[50] G. E. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *NC*, 2006.

[51] J. Hosang, R. Benenson, P. Dollár, and B. Schiele. What makes for effective detection proposals? *T-PAMI*, 2016.

[52] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

[53] S. Jégou, M. Drozdzal, D. Vazquez, A. Romero, and Y. Bengio. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. *CVPR*, 2017.

[54] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, and M. Song. Neural style transfer: A review. *VCG*, 2019.

[55] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. *ECCV*, 2016.

[56] Y. Ke, R. Sukthankar, et al. Pca-sift: A more distinctive representation for local image descriptors. *CVPR*, 2004.

[57] A. Khoreva, R. Benenson, J. H. Hosang, M. Hein, and B. Schiele. Simple does it: Weakly supervised instance and semantic segmentation. *CVPR*, 2017.

[58] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *ICLR*, 2015.

[59] A. Kirillov, E. Levinkov, B. Andres, B. Savchynskyy, and C. Rother. Instancecut: from edges to instances with multicut. *CVPR*, 2017.

[60] A. Kolesnikov and C. H. Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. *ECCV*, 2016.

[61] S. Kong and C. Fowlkes. Recurrent pixel embedding for instance grouping. *CVPR*, 2018.

[62] T. K. Konopczynski, T. Kröger, L. Zheng, and J. Hesser. Instance segmentation of fibers from low resolution ct scans via 3d deep embedding learning. *BMVC*, 2018.

[63] S. Kumagai, K. Hotta, and T. Kurita. Mixture of counting cnns: Adaptive integration of cnns specialized to specific appearance for crowd counting. *arXiv*, 2017.

[64] M. Lapin, M. Hein, and B. Schiele. Analysis and optimization of loss functions for multiclass, top-k, and multilabel classification. *PAMI*, 2018.

[65] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt. Where are the blobs: Counting by localization with point supervision. *ECCV*, 2018.

[66] I. H. Laradji, N. Rostamzadeh, P. O. Pinheiro, D. Vazquez, and M. Schmidt. Instance segmentation with point supervision. *arXiv*, 2019.

[67] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *IEEE*, 1998.

[68] V. Lempitsky and A. Zisserman. Learning to count objects in images. *NIPS*, 2010.

[69] D. Li, J.-B. Huang, Y. Li, S. Wang, and M.-H. Yang. Weakly supervised object localization with progressive domain adaptation. *CVPR*, 2016.

[70] M. Li, Z. Zhang, K. Huang, and T. Tan. Estimating the number of people in crowded scenes by mid based foreground segmentation and head-shoulder detection. *ICPR*, 2008.

[71] Y. Li, X. Zhang, and D. Chen. Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes. *CVPR*, 2018.

[72] X. Liang, Y. Wei, X. Shen, J. Yang, L. Lin, and S. Yan. Proposal-free network for instance-level object segmentation. *arXiv*, 2015.

[73] X. Liang, Y. Wei, X. Shen, Z. Jie, J. Feng, L. Lin, and S. Yan. Reversible recursive instance-level object segmentation. *CVPR*, 2016.

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

[75] 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. *ECCV*, 2014.

[76] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. *CVPR*, 2016.

[77] S. Liu, J. Jia, S. Fidler, and R. Urtasun. Sgn: Sequential grouping networks for instance segmentation. *ICCV*, 2017.

[78] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg. Ssd: Single shot multibox detector. *ECCV*, 2016.

[79] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. *CVPR*, 2015.

[80] M. Long, Y. Cao, J. Wang, and M. I. Jordan. Learning transferable features with deep adaptation networks. *arXiv*, 2015.

[81] C. C. Loy, K. Chen, S. Gong, and T. Xiang. Crowd counting and profiling: Methodology and evaluation. *MSVAC*, 2013.

[82] K.-K. Maninis, J. Pont-Tuset, P. Arbeláez, and L. Van Gool. Convolutional oriented boundaries. *ECCV*, 2016.

[83] M. Marsde, K. McGuinness, S. Little, C. E. Keogh, and N. E. O'Connor. People, penguins and petri dishes: adapting object counting models to new visual domains and object types without forgetting. *CVPR*, 2018.

[84] F. Massa and R. Girshick. maskrcnn-benchmark: Fast, modular reference implementation of Instance Segmentation and Object Detection algorithms in PyTorch, 2018.

[85] M. M. Najafabadi, F. Villanustre, T. M. Khoshgoftaar, N. Seliya, R. Wald, and E. Muharemagic. Deep learning applications and challenges in big data analytics. *JBD*, 2015.

[86] A. Newell, Z. Huang, and J. Deng. Associative embedding: End-to-end learning for joint detection and grouping. *NIPS*, 2017.

[87] J. Nutini, B. Sepehry, I. Laradji, M. Schmidt, H. Koepke, and A. Virani. Convergence rates for greedy kaczmarz algorithms, and faster randomized kaczmarz rules using the orthogonality graph. *arXiv*, 2016.

[88] J. Nutini, I. Laradji, and M. Schmidt. Let's make block coordinate descent go fast: Faster greedy rules, message-passing, active-set complexity, and superlinear convergence. *arXiv*, 2017.

[89] D. Onoro-Rubio and R. J. López-Sastre. Towards perspective-free object counting with deep learning. *ECCV*, 2016.

[90] P. O. Pinheiro and R. Collobert. From image-level to pixel-level labeling with convolutional networks. *CVPR*, 2015.

[91] P. O. Pinheiro, R. Collobert, and P. Dollár. Learning to segment object candidates. *NIPS*, 2015.

[92] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. *ECCV*, 2016.

[93] P. O. Pinheiro, T.-Y. Lin, R. Collobert, and P. Dollár. Learning to refine object segments. *ECCV*, 2016.

[94] R. Pohle and K. D. Toennies. Segmentation of medical images using adaptive region growing. *MIIP*, 2001.

[95] J. Pont-Tuset and L. Van Gool. Boosting object proposals: From pascal to coco. *CVPR*, 2015.

[96] X. Qi, Z. Liu, J. Shi, H. Zhao, and J. Jia. Augmented feedback in semantic segmentation under image level supervision. *ECCV*, 2016.

[97] V. Rabaud and S. Belongie. Counting crowded moving objects. *CVPR*, 2006.

[98] M. Rahnemoonfar and C. Sheppard. Deep count: fruit counting based on deep simulated learning. *Sensors*, 2017.

[99] J. Redmon and A. Farhadi. Yolov3: An incremental improvement. *arXiv*, 2018.

[100] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CVPR*, 2016.

[101] T. Remez, J. Huang, and M. Brown. Learning to segment via cut-and-paste. *ECCV*, 2018.

[102] M. Ren and R. S. Zemel. End-to-end instance segmentation with recurrent attention. *CVPR*, 2017.

[103] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NIPS*, 2015.

[104] B. Romera-Paredes and P. H. S. Torr. Recurrent instance segmentation. *ECCV*, 2016.

[105] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. *MICCAI*, 2015.

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

[107] A. Salman, A. Jalal, F. Shafait, A. Mian, M. Shortis, J. Seager, and E. Harvey. Fish species classification in unconstrained underwater environments based on deep learning. *LOM*, 2016.

[108] P. Samangouei, M. Kabkab, and R. Chellappa. Defense-gan: Protecting classifiers against adversarial attacks using generative models. *arXiv*, 2018.

[109] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, D. Batra, et al. Grad-cam: Visual explanations from deep networks via gradient-based localization. *ICCV*, 2017.

[110] B. Settles. Active learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2009.

[111] J. Shi and C. Tomasi. Good features to track. *Cornell University*, 1993.

[112] N. Silberman, D. Sontag, and R. Fergus. Instance segmentation of indoor scenes using a coverage loss. *ECCV*, 2014.

[113] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015.

[114] V. A. Sindagi and V. M. Patel. Generating high-quality crowd density maps using contextual pyramid cnns. *ICCV*, 2017.

[115] V. A. Sindagi and V. M. Patel. A survey of recent advances in cnn-based single image crowd counting and density estimation. *PRL*, 2017.

[116] H. O. Song, R. Girshick, S. Jegelka, J. Mairal, Z. Harchaoui, and T. Darrell. On learning to localize objects with minimal supervision. *arXiv*, 2014.

[117] H. O. Song, Y. J. Lee, S. Jegelka, and T. Darrell. Weakly-supervised discovery of visual pattern configurations. *NIPS*, 2014.

[118] C. Szegedy, S. Ioffe, V. Vanhoucke, and A. A. Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. *AAAI*, 2017.

[119] P. Tang, X. Wang, X. Bai, and W. Liu. Multiple instance detection network with online instance classifier refinement. *CVPR*, 2017.

[120] P. Tang, X. Wang, A. Wang, Y. Yan, W. Liu, J. Huang, and A. Yuille. Weakly supervised region proposal network and object detection. *ECCV*, 2018.

[121] P. Tu, T. Sebastian, G. Doretto, N. Krahnstoever, J. Rittscher, and T. Yu. Unified crowd segmentation. *ECCV*, 2008.

[122] O. Tuzel, F. Porikli, and P. Meer. Pedestrian detection via classification on riemannian manifolds. *PAMI*, 2008.

[123] J. Uhrig, M. Cordts, U. Franke, and T. Brox. Pixel-level encoding and depth layering for instance-level semantic labeling. *GCPR*, 2016.

[124] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *ICCV*, 2013.

[125] E. Walach and L. Wolf. Learning to count with cnn boosting. *ECCV*, 2016.

[126] F. Wan, C. Liu, W. C. Ke, X. Ji, J. Jiao, and Q. Ye. C-mil: Continuation multiple instance learning for weakly supervised object detection. *arXiv*, 2019.

[127] L. Wang, J. Shi, G. Song, and I.-f. Shen. Object detection combining recognition and segmentation. *ACCV*, 2007.

[128] M. Wang and X. Wang. Automatic adaptation of a generic pedestrian detector to a specific traffic scene. *CVPR*, 2011.

[129] X. Wang, X. Ma, and W. E. L. Grimson. Unsupervised activity perception in crowded and complicated scenes using hierarchical bayesian models. *PAMI*, 2009.

[130] Y. Wang and Q. Yao. Few-shot learning: A survey. *arXiv*, 2019.

[131] K. Wu, E. Otoo, and A. Shoshani. Optimizing connected component labeling algorithms. *MIIP*, 2005.

[132] G. Zen, N. Rostamzadeh, J. Staiano, E. Ricci, and N. Sebe. Enhanced semantic descriptors for functional scene categorization. *ICPR*, 2012.

[133] K. Zhang, W.-L. Chao, F. Sha, and K. Grauman. Video summarization with long short-term memory. *ECCV*, 2016.

[134] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura. Fcn-rlstm: Deep spatio-temporal neural networks for vehicle counting in city cameras. *ICCV*, 2017.

[135] S. Zhang, G. Wu, J. P. Costeira, and J. M. Moura. Understanding traffic density from large-scale web camera data. *CVPR*, 2017.

[136] Y. Zhang, D. Zhou, S. Chen, S. Gao, and Y. Ma. Single-image crowd counting via multi-column convolutional neural network. *CVPR*, 2016.

[137] Z. Zhang, A. G. Schwing, S. Fidler, and R. Urtasun. Monocular object instance segmentation and depth ordering with cnns. *CVPR*, 2015.

[138] Z. Zhang, S. Fidler, and R. Urtasun. Instance-level segmentation for autonomous driving with deep densely connected mrfs. *CVPR*, 2016.

[139] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. *CVPR*, 2017.

[140] B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *CVPR*, 2016.

[141] Y. Zhou, Y. Zhu, Q. Ye, Q. Qiu, and J. Jiao. Weakly supervised instance segmentation using class peak response. *CVPR*, 2018.

[142] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao. Soft proposal networks for weakly supervised object localization. *ICCV*, 2017.

[143] C. L. Zitnick and P. Dollár. Edge boxes: Locating object proposals from edges. *ECCV*, 2014.