Where Are the Blobs: Counting by Localization with Point Supervision

Issam H. Laradji[1,2(✉)], Negar Rostamzadeh[1], Pedro O. Pinheiro[1], David Vazquez[1], and Mark Schmidt[1,2]

[1] Element AI, Montreal, Canada
{negar,pedro,dvazquez}@elementai.com
[2] Department of Computer Science, University of British Columbia, Vancouver, Canada
{issamou,schmidtm}@cs.ubc.ca

Abstract. 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.

1 Introduction

Object counting is an important task in computer vision with many applications in surveillance systems [3,4], traffic monitoring [5,6], ecological surveys [1], and cell counting [7,8]. 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, it has been used for counting objects present in everyday scenes in challenging datasets where the objects of interest come from a large number of classes such as the Pascal VOC dataset [2].

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. 1). Also, the background, weather conditions, and

V. Ferrari et al. (Eds.): ECCV 2018, LNCS 11206, pp. 560–576, 2018.
https://doi.org/10.1007/978-3-030-01216-8_34

Fig. 1. Qualitative results on the Penguins [1] and PASCAL VOC datasets [2]. 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. (Color figure online)

illuminations can vary widely across the scenes. Therefore, the model needs to be robust enough 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 [9,10]. 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 [9]. 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. 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 [11–13]. 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 [14]. 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 [6] 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) enforces it to predict the semantic segmentation labels for each pixel in the image (similar to [14]) 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 [15], UCSD [16], and ShanghaiTech B [17] as crowd datasets; MIT Traffic [18], and Park lot [5] as surveillance datasets; Trancos [6] as a traffic monitoring dataset; and Penguins [1] as a population monitoring dataset. We also show counting results for the PASCAL VOC [2] 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.

The rest of the paper is organized as follows: Sect. 2 presents related works on object counting; Sect. 3 describes the proposed approach; and Sect. 4 describes our experiments and results. Finally, we present the conclusion in Sect. 5.

2 Related Work

Object counting has received significant attention over the past years [8,9,19]. It can be roughly divided into three categories [20]: (1) counting by clustering, (2) counting by regression, and (3) counting by detection.

Early work in object counting use *clustering-based methods*. They are unsupervised approaches where objects are clustered based on features such as appearance and motion cues [19,21]. Rabaud and Belongie [19] proposed to use feature points which are detected by motion and appearance cues and are tracked through time using KLT [22]. The objects are then clustered based on similar features. Sebastian *et al.* [21] 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 *et al.* [8] 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.* [17] 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 *et al.* [10] 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' [9], which explicitly learns to count using image-level labels only. Glance methods are efficient if the object count is small [9]. 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 [11,12,23] and per-pixel labels [24–26] ground-truth. Perfect object detection implies perfect count. However, Chattopadhyay *et al.* [9] showed that Fast RCNN [27], 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 [8–10] 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.* [28–30]. 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 [8].

Our method is also similar to segmentation methods such as U-net [31] 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.

3 Localization-Based Counting FCN

Our model is based on the fully convolutional neural network (FCN) proposed by Long *et al.* [24]. 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.

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, enforces 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.* [14]. 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 Sect. 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. 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:

$$\mathcal{L}(S,T) = \underbrace{\mathcal{L}_I(S,T)}_{\text{Image-level loss}} + \underbrace{\mathcal{L}_P(S,T)}_{\text{Point-level loss}} + \underbrace{\mathcal{L}_S(S,T)}_{\text{Split-level loss}} + \underbrace{\mathcal{L}_F(S,T)}_{\text{False positive loss}} , \quad (1)$$

which we describe in detail next.

Image-Level Loss. 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:

$$\mathcal{L}_I(S,T) = -\frac{1}{|C_e|}\sum_{c \in C_e} \log(S_{t_c c}) - \frac{1}{|C_{\neg e}|}\sum_{c \in C_{\neg e}} \log(1 - S_{t_c c}) , \quad (2)$$

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. $\mathcal{I}_s$ represents the locations of the object instances. This is formally defined as,

$$\mathcal{L}_P(S,T) = -\sum_{i \in \mathcal{I}_s} \log(S_{iT_i}) \ , \tag{3}$$

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. These splits are made by first finding boundaries between object pairs. The model then learns to predict these boundaries as the background class. 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 [32] 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),

1. *Line split method.* 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,
$$z(E) = \frac{1}{|E|} \sum_{i \in E} S_{i0} \ , \tag{4}$$
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 [33] globally on the input image where we set the ground-truth 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.

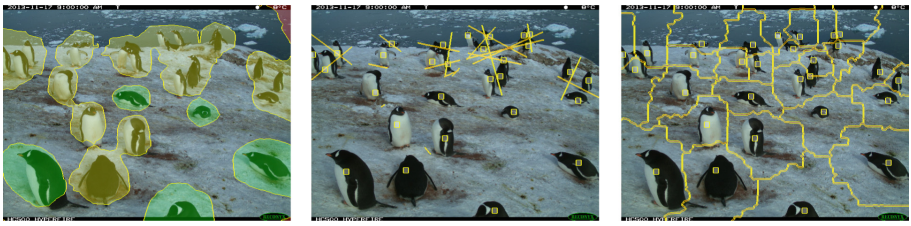

Fig. 2. Split methods. Comparison between the line split, and the watershed split. The loss function identifies the boundary splits (shown as yellow lines). Yellow blobs represent those with more than one object instance, and red blobs represent those that have no object instance. Green blobs are true positives. The squares represent the ground-truth point annotations. (Color figure online)

Figure 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,

$$\mathcal{L}_S(S, T) = -\sum_{i \in T_b} \alpha_i \log(S_{i0}), \tag{5}$$

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 the boundaries 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 [34,35] 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. $\mathcal{L}_F$ discourages the model from predicting a blob with no point annotations, in order to reduce the number of false positive predictions. The loss function is defined as

$$\mathcal{L}_F(S, T) = -\sum_{i \in B_{fp}} \log(S_{i0}), \tag{6}$$

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

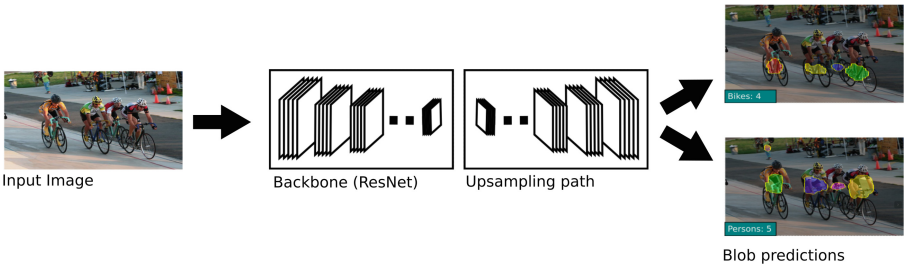

Fig. 3. Given an input image, our model first extracts features using a backbone architecture such as ResNet. The extracted features are then upsampled through the upsampling path to obtain blobs for the objects. In this example, the model predicts the blobs for persons and bikes for an image in the PASCAL VOC 2007 dataset.

(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. 5). Therefore, optimizing this loss term results in less false positive predictions as shown in the qualitative results in Fig. 5. The experiments show that this loss term is extremely important for accurate object counting.

3.2 LC-FCN Architecture and Inference

LC-FCN can be any FCN architecture such as FCN8 architecture [24], Deeplab [36], Tiramisu [25], and PSPnet [26]. LC-FCN consists of a backbone that extracts the image features. The backbone is an Imagenet pretrained network such as VGG16 or ResNet-50 [37,38]. 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. 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 [32] on F to get the blobs for each class c. The count is the number of predicted blobs (see Fig. 3).

4 Experiments

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

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

Table 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.

	Separated		Mixed	
Method	Max	Median	Max	Median
Density-only [1]	8.11	5.01	9.81	7.09
With seg. and depth [1]	6.38	3.99	5.74	3.42
With seg and no depth [1]	5.77	3.41	5.35	3.26
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**

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 [39,40]. For datasets with multi-class objects, we report the mean root mean square error (mRMSE) as used in [9] for the PASCAL VOC 2007 dataset. We measure the localization performance using the average mean absolute error (GAME) as in [6]. Since our model predicts blobs instead of a density map, GAME might not be an accurate localization measure. Therefore, in Sect. 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 [41] 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 [24]; ResFCN which is FCN8 that uses ResNet-50 as the backbone instead of VGG16; and PSPNet [26] with ResNet-101 as the backbone. We use the watershed split procedure in all our experiments.

4.2 Results and Discussion

Penguins Dataset [1]**.** The Penguins dataset comprises images of penguin colonies located in Antarctica. We use the two dataset splits as in [1]. 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 1, the MAE is computed with respect to the Max and Median count (as there are multiple annotators). Our methods significantly outperform theirs in all of the four settings, although their methods use depth features and the multiple annotations

Table 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 [6].

Method	MAE	GAME (1)	GAME (2)	GAME (3)
Lemptisky+SIFT [6]	13.76	16.72	20.72	24.36
Hydra CCNN [10]	10.99	13.75	16.69	19.32
FCN-MT [42]	5.31	-	-	-
FCN-HA [43]	4.21	-	-	-
CSRNet [44]	3.56	5.49	8.57	15.04
Glance	7.0	-	-	-
LC-FCN8	4.53	7.00	10.66	16.05
LC-ResFCN	**3.32**	5.2	7.92	12.57
LC-PSPNET	3.57	**4.98**	**7.42**	**11.67**

provided for each penguin. This suggests that LC-FCN can learn to distinguish between individual penguins despite the heavy occlusions and crowding.

Trancos Dataset [10]**.** 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 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, 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 [5]**.** 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 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 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

Table 3. PASCAL VOC. We compare against the methods proposed in [9]. Our model evaluates on the full test set, whereas the other methods take the mean of ten random samples of the test set evaluation.

Method	mRMSE	mRMSE-nz	m-relRMSE	m-relRMSE-nz
Glance-noft-2L [9]	0.50	1.83	0.27	0.73
Aso-sub-ft-3 × 3 [9]	0.42	1.65	0.21	0.68
Faster-RCNN [9]	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 4. Crowd datasets MAE results.

Methods	UCSD	Mall	ShanghaiTech B
FCN-rLSTM [43]	1.54	-	-
MoCNN [45]	-	2.75	-
CNN-boosting [46]	1.10	2.01	-
M-CNN [17]	1.07	-	26.4
CP-CNN [47]	-	-	20.1
CSRNet [44]	1.16	-	**10.6**
LC-FCN8	1.51	2.42	13.14
LC-ResFCN	**0.99**	2.12	25.89
LC-PSPNet	1.01	**2.00**	21.61

Fig. 4. Predicted blobs on a ShanghaiTech B test image.

image-level labels (see Fig. 5 for qualitative results). Note that this is the first counting method being applied on this dataset.

MIT Traffic [3]**.** 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 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 [2]**.** We use the standard training, validation, and test split as specified in [2]. We use the point-level annotation ground-truth provided by Bearman *et al.* [14] to train our LC-FCN methods. We evaluated against the count of the non-difficult instances of the Pascal VOC 2007 test set.

Table 3 compares the performance of LC-FCN with different methods proposed by [9]. We point the reader to [9] 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

Table 5. Quantitative results. Comparison of different parts of the proposed loss function for counting and localization performance.

Method	MIT Traffic		PKLot		Trancos		Penguins Separated	
	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	**3.32**	0.68	3.96	0.63
LC-FCN8	**0.91**	0.69	**0.21**	**0.99**	4.53	**0.54**	**3.74**	**0.61**

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.

Crowd Counting Datasets. Table 4 reports the MAE score of our method on 3 crowd datasets using the setup described in the survey paper [40]. 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. Figure 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.

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 5 shows the localization results of our method on several datasets.

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 [14]. We observe in Fig. 5(b) that a model using these two terms results in a single blob

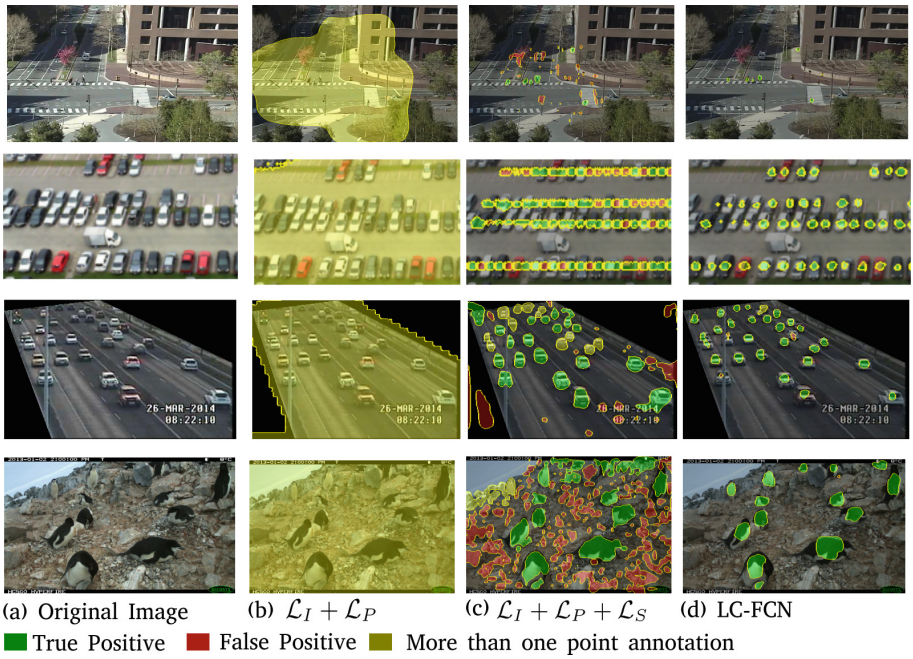

Fig. 5. Qualitative results of LC-FCN trained with different terms of the proposed loss function. (a) Test images obtained from MIT Traffic, Parking Lot, Trancos, and Penguins. (b) Prediction results using only image-level and point-level loss terms. (c) Prediction results using image-level, point-level, 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. (Color figure online)

that groups many object instances together. Consequently, this performs poorly in terms of the mean absolute error and the F-Score (see Table 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. 5(c) that a model using this additional loss term predicts several blobs as object instances rather than one large single blob. However, since $\mathcal{L}_I + \mathcal{L}_P + \mathcal{L}_S$ does not penalize the model from 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. 5(d) and Table 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 method	Trancos	Penguins
LC-ResFCN (L)	4.77	1.89
LC-ResFCN (W)	3.34	0.95

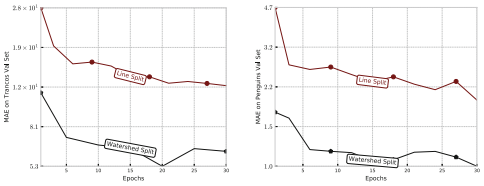

Fig. 6. Split Heuristics Analysis. Comparison between the watershed split method and the line split method against the validation MAE score.

Split Heuristics Analysis. In Fig. 6 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.

5 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. For future work, we plan to explore different FCN architectures and splitting methods that LC-FCN can use to efficiently split between overlapping objects that have complicated shapes and appearances.

Acknowledgements. We would like to thank the anonymous referees for their useful comments that significantly improved the paper. Issam Laradji is funded by the UBC Four-Year Doctoral Fellowships (4YF).

References

1. Arteta, C., Lempitsky, V., Zisserman, A.: Counting in the wild. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 483–498. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_30
2. Everingham, M., Eslami, S.M., Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes challenge: a retrospective. IJCV **111**, 98–136 (2015)
3. Wang, M., Wang, X.: Automatic adaptation of a generic pedestrian detector to a specific traffic scene. In: CVPR (2011)
4. Zen, G., Rostamzadeh, N., Staiano, J., Ricci, E., Sebe, N.: Enhanced semantic descriptors for functional scene categorization. In: ICPR (2012)

5. De Almeida, P.R., Oliveira, L.S., Britto Jr., A.S., Silva Jr., E.J., Koerich, A.L.: PKLot-a robust dataset for parking lot classification. Expert Syst. Appl. **42**, 4937–4949 (2015)
6. Guerrero-Gómez-Olmedo, R., Torre-Jiménez, B., López-Sastre, R., Maldonado-Bascón, S., Oñoro-Rubio, D.: Extremely overlapping vehicle counting. In: Paredes, R., Cardoso, J.S., Pardo, X.M. (eds.) IbPRIA 2015. LNCS, vol. 9117, pp. 423–431. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-19390-8_48
7. Cohen, J.P., Boucher, G., Glastonbury, C.A., Lo, H.Z., Bengio, Y.: Count-ception: counting by fully convolutional redundant counting. In: ICCV Workshops (2017)
8. Lempitsky, V., Zisserman, A.: Learning to count objects in images. In: NIPS (2010)
9. Chattopadhyay, P., Vedantam, R., Selvaraju, R.R., Batra, D., Parikh, D.: Counting everyday objects in everyday scenes. In: CVPR (2017)
10. Oñoro-Rubio, D., López-Sastre, R.J.: Towards perspective-free object counting with deep learning. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 615–629. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_38
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS (2015)
12. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR (2016)
13. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: CVPR (2017)
14. Bearman, A., Russakovsky, O., Ferrari, V., Fei-Fei, L.: What's the point: semantic segmentation with point supervision. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9911, pp. 549–565. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46478-7_34
15. Chen, K., Loy, C.C., Gong, S., Xiang, T.: Feature mining for localised crowd counting. In: BMVC (2012)
16. Chan, A.B., Liang, Z.S.J., Vasconcelos, N.: Privacy preserving crowd monitoring: counting people without people models or tracking. In: CVPR (2008)
17. Zhang, Y., Zhou, D., Chen, S., Gao, S., Ma, Y.: Single-image crowd counting via multi-column convolutional neural network. In: CVPR (2016)
18. Wang, X., Ma, X., Grimson, W.E.L.: Unsupervised activity perception in crowded and complicated scenes using hierarchical Bayesian models. PAMI **31**, 539–555 (2009)
19. Rabaud, V., Belongie, S.: Counting crowded moving objects. In: CVPR (2006)
20. Loy, C.C., Chen, K., Gong, S., Xiang, T.: Crowd counting and profiling: methodology and evaluation. In: Ali, S., Nishino, K., Manocha, D., Shah, M. (eds.) Modeling, Simulation and Visual Analysis of Crowds. TISVC, vol. 11, pp. 347–382. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-8483-7_14
21. Tu, P., Sebastian, T., Doretto, G., Krahnstoever, N., Rittscher, J., Yu, T.: Unified crowd segmentation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008. LNCS, vol. 5305, pp. 691–704. Springer, Heidelberg (2008). https://doi.org/10.1007/978-3-540-88693-8_51
22. Shi, J., Tomasi, C.: Good Features to Track. Cornell University, Ithaca (1993)
23. Liu, W., et al.: SSD: single shot multibox detector. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 21–37. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_2
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: CVPR (2015)

25. Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y.: The one hundred layers tiramisu: fully convolutional densenets for semantic segmentation. In: CVPR (2017)
26. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: CVPR (2017)
27. Girshick, R.: Fast R-CNN. In: ICCV (2015)
28. Arteta, C., Lempitsky, V., Noble, J.A., Zisserman, A.: Learning to detect cells using non-overlapping extremal regions. In: Ayache, N., Delingette, H., Golland, P., Mori, K. (eds.) MICCAI 2012. LNCS, vol. 7510, pp. 348–356. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33415-3_43
29. Arteta, C., Lempitsky, V., Noble, J.A., Zisserman, A.: Learning to detect partially overlapping instances. In: CVPR (2013)
30. Arteta, C., Lempitsky, V., Noble, J.A., Zisserman, A.: Detecting overlapping instances in microscopy images using extremal region trees. MIA **27**, 3–16 (2016)
31. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
32. Wu, K., Otoo, E., Shoshani, A.: Optimizing connected component labeling algorithms. In: Image Processing, Medical Imaging (2005)
33. Beucher, S., Meyer, F.: The morphological approach to segmentation: the watershed transformation. Opt. Eng.-N. Y.-Marcel Dekker Inc. **34**, 433 (1992)
34. Nutini, J., Laradji, I., Schmidt, M.: Let's make block coordinate descent go fast: faster greedy rules, message-passing, active-set complexity, and superlinear convergence. arXiv (2017)
35. Nutini, J., Sepehry, B., Laradji, I., Schmidt, M., Koepke, H., Virani, A.: Convergence rates for greedy Kaczmarz algorithms, and faster randomized Kaczmarz rules using the orthogonality graph. arXiv (2016)
36. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. PAMI **40**, 834–848 (2018)
37. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
38. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR (2009)
39. Charles, R.M., Taylor, K.M., Curry, J.H.: Nonnegative matrix factorization applied to reordered pixels of single images based on patches to achieve structured nonnegative dictionaries. arXiv (2015)
40. Sindagi, V.A., Patel, V.M.: A survey of recent advances in CNN-based single image crowd counting and density estimation. Pattern Recognit. Lett. (2017)
41. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. arXiv (2014)
42. Zhang, S., Wu, G., Costeira, J.P., Moura, J.M.: Understanding traffic density from large-scale web camera data. In: CVPR (2017)
43. Zhang, S., Wu, G., Costeira, J.P., Moura, J.M.: FCN-rLSTM: deep spatio-temporal neural networks for vehicle counting in city cameras. In: ICCV (2017)
44. Li, Y., Zhang, X., Chen, D.: CSRNET: dilated convolutional neural networks for understanding the highly congested scenes. In: CVPR (2018)
45. Kumagai, S., Hotta, K., Kurita, T.: Mixture of counting CNNs: adaptive integration of CNNs specialized to specific appearance for crowd counting. arXiv (2017)

46. Walach, E., Wolf, L.: Learning to count with CNN boosting. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9906, pp. 660–676. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46475-6_41
47. Sindagi, V.A., Patel, V.M.: Generating high-quality crowd density maps using contextual pyramid CNNs. In: ICCV (2017)